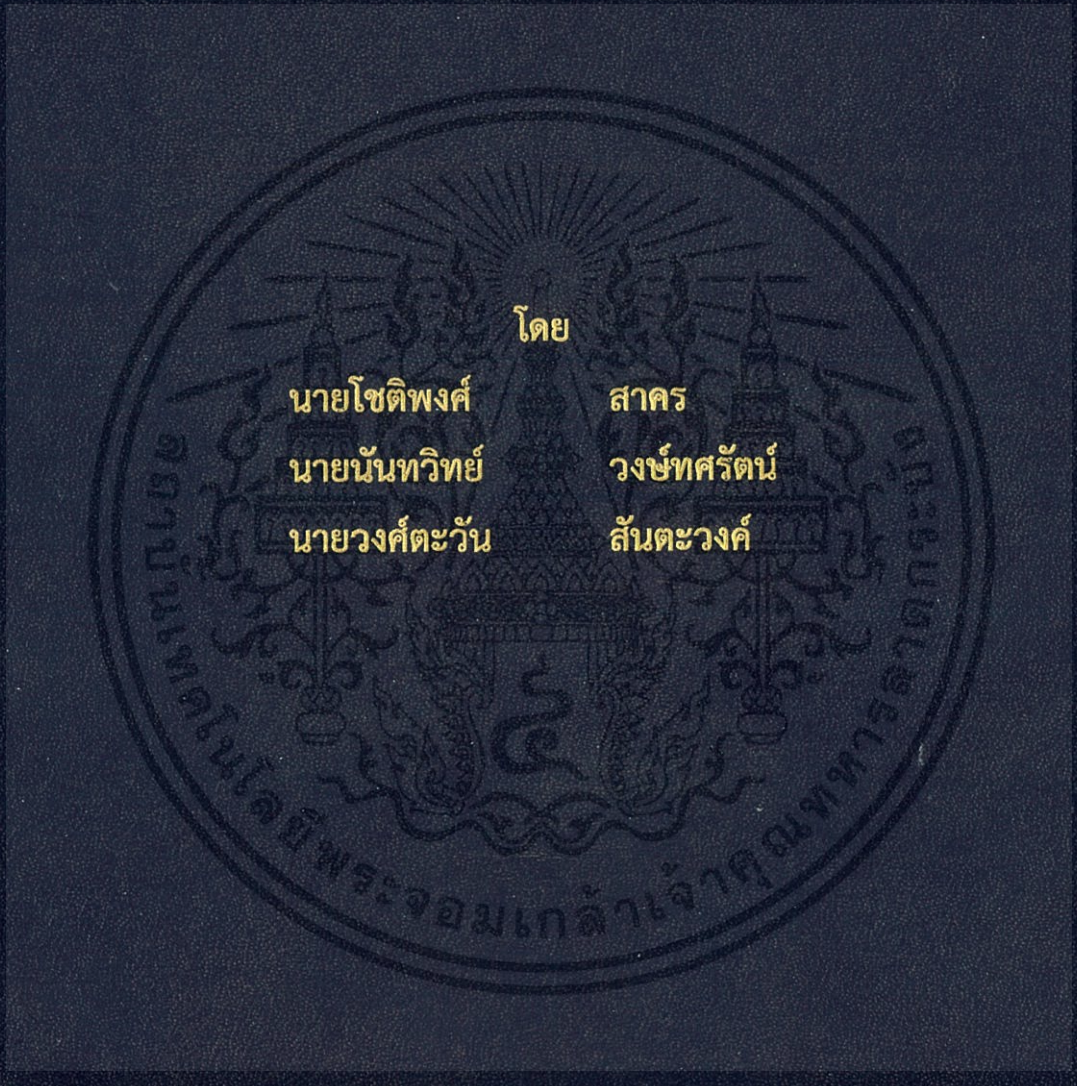
ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก รุ่นที่ 3

LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES VERSION 3



โดย

นายโชติพงศ์ สาคร

นายนันทวิทย์ วงษ์ทศรัตน์

นายวงศ์ตะวัน สันตะวงศ์

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก รุ่นที่ 3

LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES VERSION 3

โดย

นายโชติพงศ์	สาคร	รหัส 58010303
นายนันทวิทย์	วงษ์ทศรัตน์	รหัส 58010669
นายวงศ์ตะวัน	สันตะวงค์	รหัส 58011077

อาจารย์ที่ปรึกษา

ศ. ดร. พรชัย ทรัพย์นิธิ

ผศ.ดร. เวธิต ภาคย์พิสุทธิ์

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

23 / 5 / 62

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

21 / 5 / 62

โครงงานปีการศึกษา 2561

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอสเอสราคาถูก รุ่นที่ 3
LOW-COST GNSS NAVIGATION SYSTEM FOR AGRICULTURAL VEHICLES
VERSION 3

ผู้จัดทำ

1. นายโชติพงศ์ สาคร 58010303
2. นายนันทวิทย์ วงษ์ทศรัตน์ 58010669
3. นายวงศ์ตะวัน สันตะวงศ์ 58011077

..... อาจารย์ที่ปรึกษา

(ศ. ดร. พรชัย ทรัพย์นิธิ)

..... อาจารย์ที่ปรึกษา (ร่วม)

(ผศ.ดร. เวธิต ภาคย์พิสุทธิ์)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือจากอาจารย์ที่ปรึกษาและผู้ที่เกี่ยวข้องในด้านต่างๆ ทางผู้จัดทำต้องขอขอบคุณ ศ.ดร.พรชัย ทรัพย์นิธิ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งท่านได้ให้คำแนะนำ ชี้แนะแนวคิดและวิธีการแก้ปัญหาในการทำงาน รวมถึงข้อคิดเห็นต่างๆที่เป็นประโยชน์ต่อการทำปริญญานิพนธ์เป็นอย่างยิ่ง ขอขอบคุณ ดร.เวธิต ภาคย์พิสุทธิ์ อาจารย์ที่ปรึกษาร่วมปริญญานิพนธ์ ท่านได้ให้ข้อมูลและคำแนะนำเพิ่มเติมจากอาจารย์ที่ปรึกษาหลัก ซึ่งช่วยเพิ่มเติมทักษะ ประสบการณ์ และเติมเต็มความรู้ให้กับผู้จัดทำ ขอขอบคุณนักศึกษาปริญญาเอก คุณจิรภูมิ บุตรโท สำหรับข้อแนะนำและความช่วยเหลือทุกๆด้านตลอดการทำปริญญานิพนธ์

สุดท้ายนี้ ผู้จัดทำขอขอบคุณบิดามารดา ครอบครัว ที่ให้โอกาสทางศึกษา และขอขอบคุณผู้เกี่ยวข้องทุกท่านที่อาจมิได้กล่าวถึงข้างต้น ที่ให้การสนับสนุนและให้ความช่วยเหลือด้วยดีเสมอมา

นายโชติพงศ์ สาคร
นายนันทวิทย์ วงษ์ทศรัตน์
นายวงศ์ตะวัน สันตะวงค์
ผู้จัดทำ

ระบบนำร่องรถทางเกษตรกรรมโดยใช้เทคโนโลยีจีเอ็นเอส
เอสราคาถูก รุ่นที่ 3
LOW-COST GNSS NAVIGATION SYSTEM FOR
AGRICULTURAL VEHICLES VERSION 3

โดย นายโชติพงศ์ สาคร 58010303
นายนันทวิทย์ วงษ์ทศรัตน์ 58010669
นายวงศ์ตะวัน สันตะวงค์ 58011077

อาจารย์ที่ปรึกษา ศ.ดร.พรชัย ทรัพย์นิธิ
อาจารย์ที่ปรึกษาร่วม ผศ.ดร.เวธิต ภาคย์พิสุทธิ์

บทคัดย่อ

ในปัจจุบันการทำเกษตรกรรมแบบอัตโนมัติได้เข้ามาทดแทนการใช้แรงงาน โดยประยุกต์ใช้ระบบนำทางเป็นตัวควบคุมเส้นทางการเคลื่อนที่ แต่การระบุตำแหน่งรถทางเกษตรกรรมที่มีความคลาดเคลื่อนต่ำ จะมีต้นทุนที่สูงเนื่องจากการใช้เครื่องรับจีเอ็นเอสเอสแบบสองความถี่ ดังนั้นในโครงงานนี้จะทำการลดต้นทุนส่วนของการใช้เครื่องรับจีเอ็นเอสเอส โดยการเปลี่ยนไปใช้เครื่องรับแบบหนึ่งความถี่ที่มีความแม่นยำใกล้เคียงกับระบบดั้งเดิม แต่มีราคาที่ย่อมเยากว่า ระบบที่ออกแบบจะมีการใช้บอร์ดประมวลผลสัญญาณที่รับได้จากเครื่องรับจีเอ็นเอสเอออกมาเป็นตำแหน่งของเครื่องรับ และทำการแก้ไขตำแหน่งที่เกิดจากการเอียงตัวของรถผ่านเซ็นเซอร์วัดความเอียงเพื่อปรับให้เป็นตำแหน่งที่แท้จริงของตัวรถ ตำแหน่งรถที่คำนวณได้จะถูกเปรียบเทียบกับเส้นทางการเดินรถปกติ เมื่อมีความผิดพลาดเกิดขึ้น ระบบจะแจ้งเตือนผู้ใช้ผ่านทางหน้าจอเพื่อให้ผู้ควบคุมสามารถควบคุมไปตามเส้นทางที่กำหนดไว้ได้

ABSTRACT

NOWADAYS, AUTOMATIC FARMING SYSTEM HAS REPLACED THE USE OF LABOR BY USING NAVIGATION SYSTEM, BUT THE HIGH ACCURACY POSITIONING SYSTEM WILL BE EXPENSIVE DUE TO THE USE OF DUAL FREQUENCY GNSS RECEIVER. THUS, IN THIS PROJECT, THE INEXPENSIVE SINGLE FREQUENCY RECEIVER WITH SIMILAR ACCURACY LEVEL WILL BE IMPLEMENTED. THE DESIGNED SYSTEM CONSISTS OF MICROCOMPUTER THAT PROCESSES SIGNAL FROM THE RECEIVER AND DISPLAYS THE POSITION OF THE RECEIVER. GYROSCOPE IS USED TO CORRECT THE POSITIONING ERROR CAUSED BY THE TILT OF THE CAR. THE ESTIMATED POSITION WILL BE COMPARED TO THE NOMINAL PATH. IF THERE IS ANY ERROR, THE SYSTEM WILL NOTIFY THE USER FROM THE DISPLAY SYSTEM, ALLOWING THE DRIVER TO KEEP TRACK ON THE DESIGNATED PATH.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

หน้า

รูปที่

สารบัญรูป(ต่อ)

หน้า

รูปที่

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากในปัจจุบันมีการใช้ประโยชน์จากการระบุพิกัดตำแหน่ง ด้วยเทคโนโลยีจีเอ็นเอสเอสกันอย่างแพร่หลาย แต่การระบุตำแหน่งที่มีความคลาดเคลื่อนต่ำนั้น จำเป็นต้องใช้เครื่องรับจีเอ็นเอสเอสแบบสองความถี่ซึ่งมีราคาที่สูง จึงต้องอาศัยระบบ RTK (Real Time Kinematic) ที่มีสถานีฐานเป็นพิกัดอ้างอิง ทำให้การระบุตำแหน่งด้วยเครื่องรับจีเอ็นเอสเอสแบบหนึ่งความถี่ซึ่งมีราคาถูก มีความแม่นยำมากยิ่งขึ้น และสามารถนำมาประยุกต์ใช้เป็นระบบนำร่องรถทางเกษตรกรรมได้

กลุ่มของข้าพเจ้าได้เล็งเห็นถึงปัญหาของผู้ใช้ทางด้านต้นทุน ความสะดวก และผลผลิตในการเก็บเกี่ยว ก่อให้เกิดแนวคิดในการลดต้นทุนการใช้เครื่องรับจีเอ็นเอสเอส โดยการสร้างอุปกรณ์ที่ใช้เครื่องรับจีเอ็นเอสเอสแบบหนึ่งความถี่ ร่วมกับการใช้เทคนิค RTK ทำให้ได้ความแม่นยำในการระบุตำแหน่งใกล้เคียงกับการใช้เครื่องรับจีเอ็นเอสเอสชนิดสองความถี่ขึ้น ในราคาที่ย่อมเยากว่า นอกจากนี้ยังใช้เซนเซอร์วัดความเอียงคำนวณตำแหน่งเปรียบเทียบกับเส้นทางการเดินรถปกติ โดยเมื่อมีการเคลื่อนที่ออกจากเส้นทางที่กำหนดระบบจะแจ้งเตือนผู้ใช้ผ่านทางหน้าจอเพื่อให้ผู้ควบคุมรถสามารถควบคุมไปตามเส้นทางที่กำหนดไว้ได้ ส่งผลให้ให้ลดการอัดแน่นของหน้าดินสำหรับใช้ในการเพาะปลูกและเพิ่มผลผลิตแบบมวลรวมให้กับเกษตรกรอีกด้วย

1.2 วัตถุประสงค์

1. ออกแบบส่วนประสานงานกราฟฟิกผู้ใช้เพื่อแสดงรายละเอียดและอำนวยความสะดวกในการเลือกเส้นทาง รวมไปถึงรูปแบบในการทำเกษตรกรรม

2. ออกแบบ และพัฒนาระบบนำร่องรถทางเกษตรกรรมความแม่นยำสูง ให้สามารถระบุตำแหน่งได้อย่างแม่นยำ และใช้งานได้อย่างมีประสิทธิภาพ

3. ออกแบบรถทางเกษตรกรรมจำลอง เพื่อใช้ทดสอบการทำงานของระบบนำร่องทางเกษตรกรรม ตลอดจนออกแบบระบบควบคุมการเคลื่อนที่อัตโนมัติสำหรับควบคุมการเคลื่อนที่ของรถเกษตรกรรมจำลองแทนการสั่งงานจากผู้ใช้

4.ออกแบบระบบฐานข้อมูล (Database) เพื่อติดตาม และตรวจสอบระบบนำร่องทางการเกษตร

1.3 ขอบเขตของโครงงาน

1. อุปกรณ์ประมวลผลพิกัดตำแหน่ง และแสดงผลต่อส่วนประสานกราฟิกกับผู้ใช้ เพื่อนำร่องให้กับการเดินรถทางเกษตรกรรมได้อย่างมีประสิทธิภาพ

2. ออกแบบโปรแกรมสำหรับสร้างเส้นทางการเดินรถ และการติดตามตำแหน่งตลอดจนแนะนำเส้นทางตามจุดพิกัดอ้างอิงผ่านส่วนประสานกราฟิกกับผู้ใช้ เพื่อให้ผู้ใช้ตรวจสอบตำแหน่งของรถ และควบคุมการเดินรถทางเกษตรกรรมได้อย่างถูกต้อง

3. ออกแบบโปรแกรมสำหรับควบคุมทิศทางการเคลื่อนที่ของรถทางเกษตรกรรมจำลอง เพื่อให้ผู้ใช้สามารถใช้ระบบควบคุมการเดินรถอัตโนมัติ

4. ออกแบบฐานข้อมูล (Database) สำหรับเก็บข้อมูลการทำงานของระบบนำร่องทางการเกษตร และสร้างเว็บไซต์สำหรับให้ผู้ใช้งานตรวจสอบการทำงานย้อนหลัง ของระบบนำร่องทางการเกษตร

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

จากโครงงานในครั้งนี้ คณะผู้จัดทำจะกล่าวถึงทฤษฎีและหลักการต่างๆ ที่เกี่ยวข้องกับความรู้เบื้องต้นเกี่ยวกับจีเอ็นเอสเอส ระบบพิกัดในแผนที่ หลักการ RTK การหาระยะทางที่สั้นที่สุดระหว่างรถทางเกษตรกรรม และเส้นทาง รวมถึงอุปกรณ์ Raspberry Pi 3

2.1 ความรู้เบื้องต้นเกี่ยวกับจีเอ็นเอสเอส

จีเอ็นเอสเอส หรือ ระบบกำหนดตำแหน่งบนพื้นโลก (GNSS: Global Navigation Satellite System) ซึ่งจะให้บริการในการระบุตำแหน่งของผู้ใช้ที่อยู่บนพื้นผิวโลกครอบคลุมทั่วทั้งโลก ในปัจจุบันมีหลายประเทศได้ส่งระบบดาวเทียมนำร่องเป็นของตัวเองหลายระบบ โดยสามารถแสดงระบบต่างๆ ได้ดังตารางที่ 2.1

ตารางที่ 2.1 ระบบดาวเทียมนำร่องที่ใช้ในปัจจุบัน

GPS (Global Positioning System)	เป็นดาวเทียมระบบแรกของโลกที่ออกแบบโดยประเทศสหรัฐอเมริกา มีดาวเทียมทั่วโลกทั้งหมด 32 ดวง [1]
GLONASS	เป็นระบบดาวเทียมของประเทศรัสเซีย มีดาวเทียมทั่วโลกทั้งหมด 30 ดวง [2]
Galileo	เป็นระบบดาวเทียมของสหภาพยุโรป ประกอบดวยดาวเทียม 30ดวง มีดาวเทียมปฏิบัติการเพียง 11 ดวง [3]
BeiDou	เป็นดาวเทียมของสาธารณรัฐประชาชนจีน ปัจจุบันเปิดให้บริการเฉพาะโซนเอเชียเท่านั้น [4]
IRNSS (Indian Regional Navigation Satellite System)	เป็นระบบดาวเทียมระดับภูมิภาคของประเทศอินเดีย ซึ่งให้บริการเฉพาะประเทศอินเดียและประเทศใกล้เคียง [5]
QZSS	เป็นระบบดาวเทียมระดับภูมิภาคของประเทศญี่ปุ่น เช่นเดียวกับ IRNSS เปิดให้บริการเฉพาะประเทศญี่ปุ่นและ และประเทศในแถบ Asia-Oceania [6]

2.1.1 องค์ประกอบของระบบดาวเทียมรับสัญญาณ

องค์ประกอบของระบบ GPS ในการสื่อสารผ่านดาวเทียม สามารถแบ่งออกได้เป็น 3 องค์ประกอบ ได้แก่

2.1.1.1 ส่วนอวกาศ (Space Segment)

กลุ่มดาวเทียมนำร่องที่อยู่ในอวกาศทำหน้าที่ส่งคลื่นสัญญาณวิทยุมายังพื้นโลก สัญญาณดังกล่าวจะมีข้อมูลนำร่อง ซึ่งภายในตัวข้อมูลนั้นจะทำให้ส่วนผู้ใช้สามารถคำนวณระบุตำแหน่งของดาวเทียมนำร่องที่มีอยู่ในอวกาศ ไม่ใช่ตำแหน่งของผู้ใช้หรือเครื่องรับสัญญาณวิทยุบนพื้นโลก

2.1.1.2 ส่วนสถานีควบคุม (Ground Station)

สถานีควบคุมภาคพื้นดิน มีหน้าที่ในสังเกต และควบคุมการทำงานของดาวเทียมให้สามารถทำงานได้ตามปกติ รวมไปถึงตรวจวัดตำแหน่งดาวเทียมนำร่องในอวกาศ แล้วส่งค่าตำแหน่งดังกล่าวกลับไปยังดาวเทียมนำร่อง เพื่อให้ข้อมูลนำร่องที่ระบุตำแหน่งของดาวเทียมนำร่องทั้งกลุ่มมีความถูกต้องแม่นยำอยู่ตลอดเวลา

2.1.1.3 ส่วนผู้ใช้ (User Segment)

ผู้ใช้บนพื้นโลกจะใช้เครื่องรับสัญญาณวิทยุในการรับสัญญาณคลื่นวิทยุจากดาวเทียมนำร่อง จากนั้นทำการคำนวณร่วมกับค่าระยะห่างระหว่างดาวเทียมนำร่อง และเครื่องรับสัญญาณวิทยุ โดยผลลัพธ์ที่ได้จะเป็นตำแหน่งของเครื่องรับสัญญาณวิทยุหรือตำแหน่งของผู้ใช้งาน

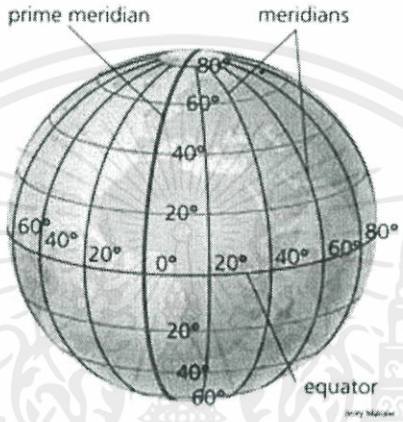
2.2 ระบบพิกัดในแผนที่

เนื่องจากโลกมีสัณฐานโดยประมาณเป็นทรงกลม เมื่อมีการกำหนดตำแหน่งพิกัดต่างๆ บนโลก จึงจำเป็นต้องฉายตำแหน่งจากพื้นที่จริงบนแผนที่ซึ่งมีลักษณะแบนราบ โดยกำหนดให้มีจุดกำเนิดของพิกัดอยู่บนผิวโลก และมีลักษณะเป็นระบบพิกัดฉาก ที่เกิดจากการตัดกันของแกนสมมติตั้งแต่ 2 แกนขึ้นไป ดังหัวข้อที่ 2.2.1 ถึงหัวข้อ 2.2.3

2.2.1 ระบบพิกัดภูมิศาสตร์ (Geographic coordinate systems) [7]

ระบบพิกัดภูมิศาสตร์เป็นระบบพิกัดที่กำหนดตำแหน่งต่างๆ บนพื้นโลก ด้วยการอ้างอิงตำแหน่งเป็นค่าระยะเชิงมุมเป็นเส้นสมมติที่เรียกว่าละติจูด (Latitude) และลองจิจูด (Longitude) ตามระยะเชิงมุมที่ห่างจากศูนย์กำเนิดของละติจูดและลองจิจูดที่ถูกกำหนดขึ้น สำหรับศูนย์กำเนิดของละติจูดนั้น กำหนดขึ้นจากแนวระดับที่ตัดผ่านศูนย์กลางของโลกและตั้งฉากกับแกนหมุน เรียกแนวระนาบศูนย์กำเนิดนั้นว่า เส้นระนาบศูนย์สูตร ซึ่งแบ่งโลกออกเป็นซีกโลกเหนือและซีกโลกใต้ ค่าระยะเชิงมุมของละติจูด จะเป็นค่าเชิงมุมที่เกิดจากมุมที่ศูนย์กลางของโลก กับแนวระดับฐานกำเนิดมุมที่เส้นระนาบศูนย์สูตร โดยวัดค่าของมุมออกไปทางซีกโลกเหนือและทางซีกโลกใต้ ค่าของมุมจะสิ้นสุดที่ขั้ว

โลกเหนือและขั้วโลกใต้ ซึ่งมีค่าเชิงมุม 90 องศา ส่วนศูนย์กำเนิดของลองจิจูดนั้น กำหนดขึ้นจากแนวระนาบทางตั้งที่ผ่านแกนหมุนของโลกตรงบริเวณตำแหน่งบนพื้นโลกที่ผ่านหอสังเกตการณ์ดาราศาสตร์เมืองกรีนิช (Greenwich) ประเทศอังกฤษ เรียกศูนย์กำเนิดนี้ว่า เส้นเมริเดียนแรก (Prime meridian) เป็นเส้นที่แบ่งโลกออกเป็นซีกโลกตะวันตกและซีกโลกตะวันออก แสดงดังรูปที่ 2.1



รูปที่ 2.1 เส้นเมริเดียนแรก (Prime meridian)

จากรูปที่ 2.1 ค่าระยะเชิงมุมของลองจิจูดเป็นค่าที่วัดมุมออกไปทางตะวันตก และตะวันออกของเส้นเมริเดียนแรก ที่วัดจากศูนย์กลางของโลกตามแนวระนาบที่มีเส้นเมริเดียนแรกเป็นฐานกำเนิดมุม ค่าของมุมจะสิ้นสุดที่เส้นเมริเดียนตรงข้ามกับเส้นเมริเดียนแรกซึ่งมีค่าของมุมซีกโลกละ 180 องศา การใช้ค่าอ้างอิงบอกตำแหน่งใช้เรียกกำหนดเช่นเดียวกับละติจูด แต่แตกต่างกันที่ต้องบอกเป็นซีกโลกตะวันตก หรือซีกโลกตะวันออก

เนื่องจากสมการที่ใช้ในการคำนวณระยะห่างระหว่างจุดแบบทั่วไป สามารถคำนวณระยะห่างระหว่างจุดได้เพียงจุดที่อยู่ในระนาบเดียวกัน ไม่สามารถคำนวณระยะทางระหว่างจุดสองจุดบนพื้นผิวโค้งได้ แต่สามารถคำนวณได้จากสมการ haversine [8] ซึ่งเป็นสมการในการหาระยะห่างโดยใช้ละติจูด และลองติจูดมาคำนวณหาระยะห่าง แสดงดังสมการที่ 2.1 2.2 และ 2.3

$$a = sin^2(\Delta\varphi/2) + cos\,\varphi_1\, cos\,\varphi_2\, sin^2(\Delta\lambda/2) \quad (2.1)$$

$$c = 2 \cdot tan^{-1}\, 2\,(\sqrt{a}, \sqrt{(1-a)}) \quad (2.2)$$

$$d = Rc \quad (2.3)$$

เมื่อ

a คือตัวแปรที่กำหนดขึ้นเพื่อนำไปใช้ในสมการที่ 2.2

c คือตัวแปรที่กำหนดขึ้นเพื่อนำไปใช้ในสมการที่ 2.3

φ_1 คือลองจิจูดตำแหน่งที่ 1

φ_2 คือลองจิจูดตำแหน่งที่ 2

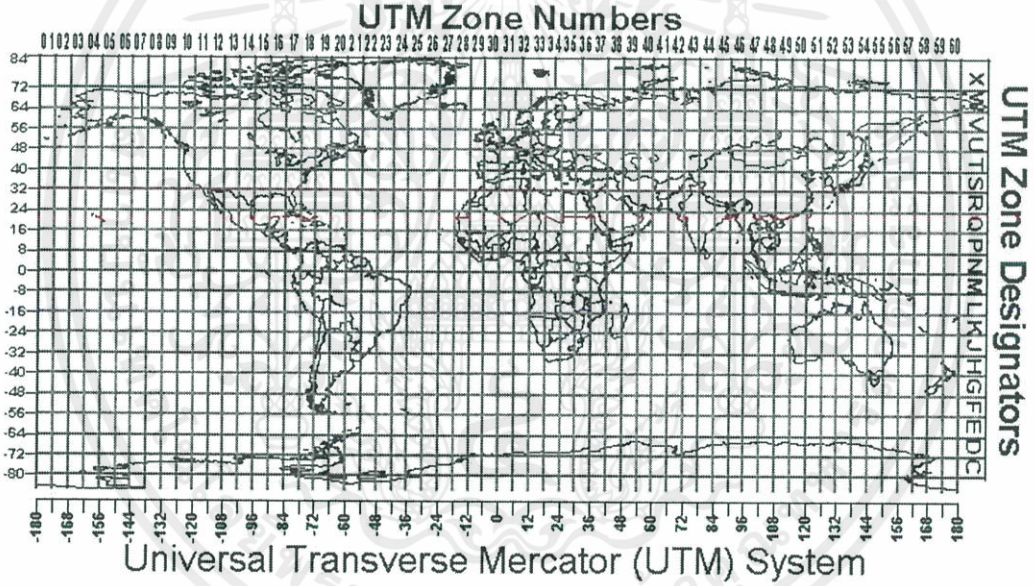
$\Delta\varphi$ คือผลต่างลองจิจูด (เรเดียน)

$\Delta\lambda$ คือผลต่างละติจูด (เรเดียน)

d คือระยะทางระหว่างจุดสองจุดบนพื้นผิวโลก (เมตร)

R คือรัศมีโลกเท่า 6,371 (กิโลเมตร)

2.2.2 ระบบพิกัดยูทีเอ็ม (UTM coordinate systems) [9]

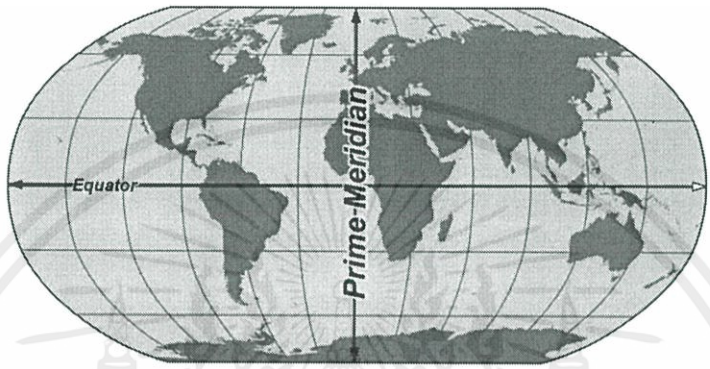


รูปที่ 2.2 แผนภาพโลกในพิกัด UTM

ระบบพิกัดยูทีเอ็ม เป็นระบบที่ปรับมาจากระบบเส้นโครงแผนที่แบบทรานสเวิร์สเมอร์เคเตอร์ ดังรูปที่ 2.2 เพื่อเป็นการรักษารูปร่างโดยใช้ทรงกระบอกตัดลูกโลกระหว่างละติจูด 84 องศาเหนือ ถึง 80 องศาใต้ โดยมีรัศมีทรงกระบอกสั้นกว่ารัศมีของลูกโลก ผิวทรงกระบอกจะผ่านเข้าไปตามแนวเมริเดียนของโซน 2 แนว คือ ตัดเข้ากับตัดออกเรียกซึ่งเรียกลักษณะนี้ว่า เส้นตัด (Secant) ทำให้ความถูกต้องมีมากขึ้น

ระบบพิกัดชนิดนี้ได้นำมาใช้ในปี ค.ศ. 1946 เพื่อให้ได้แผนที่ที่มีความละเอียดถูกต้องมากยิ่งขึ้น ระบบชนิดนี้ได้มาจากการฉายแผนที่แบบคงทิศทางเพื่อรักษารูปร่าง และมีข้อกำหนดในรายละเอียดต่างๆ ให้ถือเป็นเกณฑ์มาตรฐานเพื่อใช้งานครอบคลุมได้ทั่วโลก กำหนดให้ใช้หน่วยวัด

ระยะทางเป็นเมตร ระบบพิกัดยูทีเอ็ม ปัจจุบันเป็นที่ยอมรับกันอย่างแพร่หลายทั้งในกิจการทหารและกิจการพลเรือน สำหรับประเทศไทย รัฐบาลไทยกับรัฐบาลสหรัฐอเมริกาได้ทำข้อตกลงทำแผนที่ภายในประเทศเมื่อปี พ.ศ. 2493



รูปที่ 2.3 แสดงเส้นศูนย์สูตร และเส้นเมริเดียน

พื้นที่ของโลกระหว่างละติจูดที่ 80 องศาใต้ ถึงละติจูดที่ 84 องศาเหนือ ถูกแบ่งออกเป็นเขต (Zone) เขตละ 6 องศา รวมเป็น 60 เขต (Zone) ตามแนวลองจิจูดโดยมีหมายเลขกำกับโซนตั้งแต่ 1 ถึง 60 ตามลำดับ โดยโซนที่ 1 อยู่ระหว่างลองจิจูด 180 องศาตะวันตก ถึง 174 องศาตะวันตก โซนที่ 2 ก็อยู่ถัดไปทางด้านตะวันออกตามลำดับจนถึงโซนที่ 60 ซึ่งอยู่ระหว่างลองจิจูด 174 องศาตะวันออก ถึง 180 องศาตะวันออก และประชิดติดกับโซนที่ 1 โดยในแต่ละโซนจะมีเมริเดียนกลาง (Central meridian) เป็นของตนเอง

พื้นที่ในแต่ละโซนถูกแบ่งย่อยให้เป็นขอบเขตสี่เหลี่ยมดังรูปที่ 2.2 โดยแนวเส้นขนานละติจูดช่วงละ 8 องศา เริ่มจากเส้นขนานละติจูด 80 องศาใต้ แบ่งทีละ 8 องศา ผ่านเส้นระนาบศูนย์สูตรไปจนถึงเส้นขนานละติจูด 72 องศาเหนือ และจากเส้นขนานละติจูด 72 ถึง 84 องศาเหนือ แบ่งออกเป็นช่องละ 12 องศา รวมทั้งหมดแบ่งได้ 20 ช่องพื้นที่สี่เหลี่ยมเหล่านี้เรียกว่า เขตกริด (Grid zone) ซึ่งมีทั้งหมด 1,200 โซน การแบ่งวิธีนี้ทำให้เกิดสี่เหลี่ยมผืนผ้าเขตกริดขนาด 6 องศา x 8 องศา ยกเว้นช่วงระหว่างเส้นขนานละติจูด 72 ถึง 84 องศาเหนือ มีขนาดเขตกริดเท่ากับ 6 องศา x 12 องศา เมื่อแบ่งเสร็จแล้วได้กำหนดอักษรโรมันกำกับไว้ตั้งแต่ C ถึง X (ยกเว้น I กับ O) โดยเริ่มกำหนดอักษร C ตั้งแต่โซนของละติจูด 80 องศาใต้

การแบ่งตารางเขตกริดเหล่านี้ จะมีเลขอักษรประจำโซนของกริด (UTM Grid zone destination) โดยการอ่านหมายเลขไปทางขวาแล้วอ่านขึ้น เช่น “47 Q” หมายถึง เลขกำกับโซนในแนวตั้งที่ 47 และอักษรกำกับโซนในแนวนอนที่ Q สำหรับอักษร A, B และ Y, Z ใช้สำหรับกำกับในยูนิ

เวอร์ซัลโพลาร์สเตริโอกราฟิก (Universal Polar Stereographic: UPS) บริเวณขั้วโลกเหนือ และขั้วโลกใต้

ระบบพิกัดยูทีเอ็มใช้หน่วยระยะทางเป็นเมตร โดยในแต่ละโซนเส้นเมริเดียนกลางตัดกับเส้นระนาบศูนย์สูตรเป็นมุมฉาก ณ จุดตัดนี้เรียกว่า จุดกำเนิดโซนของระบบพิกัดยูทีเอ็ม ทิศทางที่ขนานกับแนวเมริเดียนกลาง และชี้ขึ้นไปทางเหนือ เรียกว่า ทิศเหนือกริด มีการกำหนดค่าพิกัดตะวันออกให้เส้นเมริเดียนกลางเป็น 500,000 เมตร (Easting 500,000 m.) ห่างจากจุดกำเนิดสมมติ (False origin) และกำหนดให้พิกัดเหนือสำหรับเส้นระนาบศูนย์สูตรไว้เป็น 2 กรณี สำหรับซีกโลกเหนือให้มีค่าเป็น 0 เมตร (Northing 0 m.) ห่างจากเส้นระนาบศูนย์สูตร ส่วนบริเวณใต้เส้นระนาบศูนย์สูตรมีค่าเป็น 10,000,000 เมตร (Northing 10,000,000 m.) ห่างจากจุดกำเนิดสมมติ ดังนั้นจุดศูนย์กำเนิดโซนของระบบพิกัดนี้ จึงมีค่าพิกัดเป็น E 500,000 m ; N 0 m สำหรับการใช้งานในซีกโลกเหนือและ E 500,000 m. N 10,000,000 m. สำหรับซีกโลกใต้ นอกจากนี้ขอบเขตการใช้ค่าพิกัดยูทีเอ็มสามารถเหลื่อมเข้าไปในโซนข้างเคียงได้เป็นพื้นที่กว้าง 40 กิโลเมตร เพื่อการใช้งานบริเวณขอบของโซน

2.2.3 ระบบพิกัดแผนที่ GLO (General Land Office grid system) [10]

เป็นระบบพิกัดแผนที่ที่ใช้ในการแบ่งพื้นที่สำรวจเพื่อจัดทำแผนที่ภูมิประเทศ มักใช้ในการอ่านและการทำแผนที่ธรณีวิทยา ระบบพิกัดนี้มีการแบ่งพื้นที่ออกเป็นส่วนๆ โดยมีความหมายในแต่ละส่วนดังนี้

เส้นฐานและเส้นเขตเมือง (Base line and Township line) ในบริเวณที่สำรวจเส้นละติจูดที่ใช้ในการอ้างอิงเรียกว่าเส้นฐาน เส้นขนานเหนือและใต้เส้นฐานในระยะห่างกันทุก 6 ไมล์ คือ เส้นเขตเมือง

เส้นเมริเดียนหลักและเส้นพิสัย (Principal meridian and Range line) เส้นลองจิจูดที่ใช้อ้างอิงในการสำรวจ เรียกว่า เส้นเมริเดียนหลัก จุดที่ตัดกับเส้นฐานเรียกว่า จุดเริ่มต้น (Initial point) เส้นที่ลากขนานกับเส้นเมริเดียนหลัก ไปทางตะวันออกและตะวันตกในระยะห่างทุก 6 ไมล์ คือ เส้นพิสัย

เขตเมือง คือ พื้นที่สีเหลี่ยมจัตุรัสกว้างด้านละ 6 ไมล์ ซึ่งถูกล้อมรอบด้วยเส้นเขตเมืองและเส้นพิสัย พื้นที่ 36 ตารางไมล์ กำหนดได้โดยใช้ตำแหน่งซึ่งห่างจากเส้นฐาน และเส้นเมริเดียนหลัก เช่น 2N., R.1W. อยู่ในเส้นเขตเมือง ที่ 2 เหนือจาก เส้นฐาน และเส้นพิสัย ที่ 1 ตะวันตกของเส้นเมริเดียนหลัก

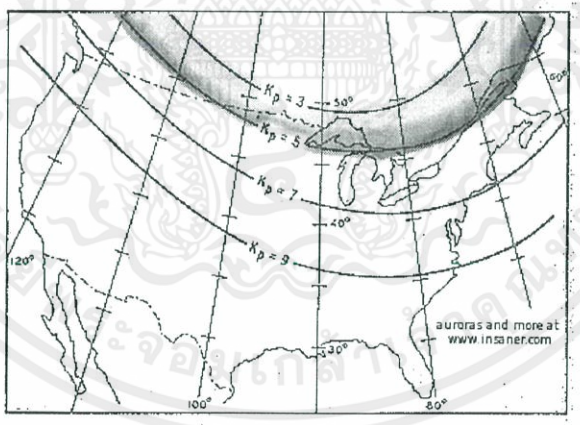
ส่วนย่อย (Section) พื้นที่ 36 ตารางไมล์ ของเส้นเขตเมืองแบ่งออกเป็นรูสี่เหลี่ยมจัตุรัส 36 รูป มีพื้นที่รูปละ 1 ตารางไมล์ พื้นที่ 1 ตารางไมล์นี้เรียกว่า ส่วนย่อย

แผนที่รูปสี่เหลี่ยม (Quadrangle) แผนที่ภูมิประเทศซึ่งแบ่งตามระบบนี้ โดยปกติเป็นรูปสี่เหลี่ยมผืนผ้า พื้นที่ของแผนที่รูปสี่เหลี่ยมล้อมรอบด้วยลองจิจูดทางทิศตะวันออกและตะวันตก และละติจูดทางทิศเหนือและใต้ ชื่อของแผนที่รูปสี่เหลี่ยมเรียกตามชื่อเมืองสำคัญ หรือลักษณะภูมิประเทศที่เด่นในแผนที่ฉบับนั้น แผนที่รูปสี่เหลี่ยมที่ใช้จะถูกแบ่งออกตามระยะห่างระหว่างลองจิจูดและละติจูดล้อมรอบอยู่เป็น 4 ชนิด คือ

แผนที่ชุด	1	องศา	(1 Degree series)	ใช้มาตราส่วน	1: 250,000
แผนที่ชุด	30	ลิปดา	(30 Minute series)	ใช้มาตราส่วน	1: 125,000
แผนที่ชุด	15	ลิปดา	(15 Minute series)	ใช้มาตราส่วน	1: 62,500
แผนที่ชุด	7.5	ลิปดา	(7.5 Minute series)	ใช้มาตราส่วน	1: 24,000

2.3 KP INDEX [11]

KP INDEX เป็นค่าที่ใช้วัดการสั่นสะเทือนของสนามแม่เหล็กโลก โดยทำการวัดจากลักษณะการสั่นสะเทือนของสนามแม่เหล็กโลกจากด้านกลางคืนของโลกที่โดนพายุสุริยะ หลักการนี้สามารถนำมาทำนายการเกิดแสงเหนือ แสดงดังรูปที่ 2.4

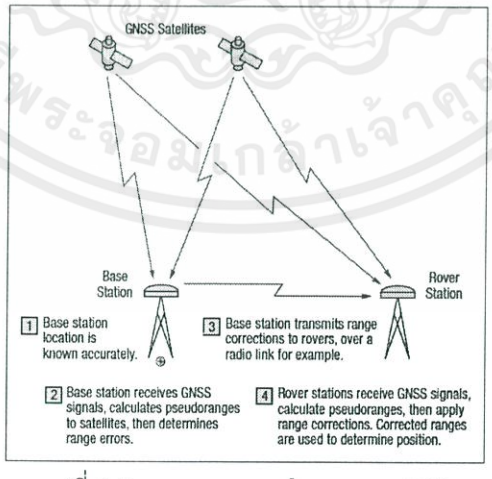


รูปที่ 2.4 การวัดแรงสั่นสะเทือนของสนามแม่เหล็ก [11]

จากรูปที่ 2.4 ระดับของการสั่นสะเทือนถูกแบ่งออกเป็นระดับ โดยมีค่าอยู่ระหว่าง 0 ถึง 9 (0 หมายถึง ไม่มีการสั่นสะเทือน และ 9 หมายถึง การสั่นสะเทือนสูงสุด)

2.4 หลักการประมาณตำแหน่งโดยใช้เทคนิค RTK

หลักการ Real Time Kinematic(RTK) เป็นการรังวัดแบบจลน์ และแสดงผลลัพธ์เป็นค่าพิกัดตำแหน่งได้ทันทีที่ทำการรังวัด ซึ่งมีวิธีการทำงานคือ ต้องใช้เครื่องรับสัญญาณดาวเทียมจีเอ็นเอสเอสอย่างน้อยสองเครื่อง โดยเครื่องที่หนึ่งจะถูกวางไว้บนหมุดที่ทราบค่าพิกัด โดยในที่นี้จุดอ้างอิงที่ใช้ในโครงงานคือ สถานีฐาน (Base Station) ที่ติดตั้งอยู่บริเวณดาดฟ้าอาคารเรียนรวม 12 ชั้น คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ส่วนอีกเครื่องจะถูกนำไปวาง ณ จุดที่ต้องการทราบค่าพิกัด และต้องมีอุปกรณ์สื่อสารเช่น ระบบ internet หรือ ระบบ Wi-Fi เป็นต้น สำหรับการรับส่งข้อมูลระหว่างสถานีฐานและสถานีผู้ใช้ เทคนิคการรังวัดแบบจลน์ในทันทีนี้มีข้อดี คือ ใช้เวลาในการรังวัดค่อนข้างเร็ว และการประมวลผลข้อมูลจะกระทำพร้อมกันขณะทำการรังวัด ทำให้ได้ค่าพิกัดตำแหน่งในทันที โดยมีความถูกต้องทางตำแหน่งในระดับเซนติเมตร แต่เทคนิคการรังวัดนี้มีข้อจำกัดคือ ความถูกต้องทางตำแหน่ง (Accuracy) และความน่าเชื่อถือของค่าพิกัด (Reliability) จะลดลงเมื่อระยะระหว่างสถานีฐานและสถานีผู้ใช้งานเพิ่มขึ้น อีกทั้งพื้นที่ขอบเขตในการทำงานได้ของแต่ละสถานีฐานไม่ต่อเนื่องกัน เพื่อลดข้อจำกัดดังกล่าว เทคนิคการรังวัดแบบจลน์ในทันทีจึงได้ถูกพัฒนาขึ้นเป็นเทคนิคการรังวัดด้วยดาวเทียมจีเอ็นเอสเอสระบบเครือข่ายแบบจลน์ในทันที (Network-based RTK GPS) ซึ่งมีข้อดีคือ มีความถูกต้องทางตำแหน่งในระดับเซนติเมตรเช่นกัน และมีความน่าเชื่อถือของค่าพิกัดสูงโดยความถูกต้องและความน่าเชื่อถือของค่าพิกัดตลอดจนขอบเขตในการทำงานนั้นเป็นอันหนึ่งอันเดียวกันตลอดทั้งโครงข่ายจีเอ็นเอสเอส ทำให้มีพื้นที่ในการทำงานเพิ่มมากขึ้น โดยอธิบายแผนภาพการทำงานของ RTK ดังรูปที่ 2.5



รูปที่ 2.5 แผนภาพการทำงานของ RTK

RTK (Real Time Kinematic) เป็นเทคนิคที่ใช้ในการระบุตำแหน่งของสถานีจลน์ (Rover Station) ด้วยการเทียบจำนวนลูกคลื่นของสัญญาณดาวเทียมที่เครื่องรับสัญญาณจีเอ็นเอสเอสของสถานีฐาน (Base Station) นับได้ นำมาหาค่าแก้ของจำนวนลูกคลื่นที่นับได้ แล้วส่งไปให้สถานีจลน์ทำการคำนวณร่วมจำนวนคลื่นที่ตัวมันเองรับได้ ก็จะได้ตำแหน่งที่ถูกต้องของสถานีจลน์ออกมา เทคนิคนี้ มีข้อจำกัดคือ ต้องทราบพิกัดของสถานีฐานอย่างแน่ชัดในบริเวณที่จะใช้เทคนิคนี้ ซึ่งจะขึ้นอยู่กับคุณภาพของสายอากาศที่ใช้รับสัญญาณจากดาวเทียม ประสิทธิภาพของหน่วยประมวลผลในแต่ละสถานี รวมไปถึงสภาพแวดล้อมในบริเวณที่ทำ RTK ควรก่อให้เกิดการรบกวนจากสัญญาณพหุวิถีน้อยที่สุด

สมการที่อธิบายถึงเทคนิคการทำงานของ RTK Carrier-phase ดังสมการที่ 2.4 และ Phase-range measurement model ดังสมการที่ 2.5

$$\begin{aligned}\phi_{r,j}^{S} &= \phi_{r,j}(t_r) - \phi_i(t^S) + \varepsilon_\phi \\ &= \left(f_i\left(t_r + dt_r(t_r) - t_0\right) + \phi_{r,0,i}\right) - \left(f_i\left(t^S + dT^S(t^S) - t_0\right) + \phi_{0,i}^{S}\right) + N_{r,i}^{S} + \varepsilon_\phi \\ &= \frac{c}{\lambda_i}\left(t_r - t^S\right) + \frac{c}{\lambda_i}\left(dt_r(t_r) - dT^S(t^S)\right) + \left(\phi_{r,0,i} - \phi_{0,j}^{S} + N_{r,i}^{S}\right) + \varepsilon_\phi \end{aligned} \tag{2.4}$$

$$\begin{aligned}\Phi_{r,j}^{S} &= \lambda_i \phi_{r,j}^{S} \\ &= c\left(t_r - t^S\right) + c\left(dt_r(t_r) - dT^S(t^S)\right) + \lambda_i\left(\phi_{r,0,i} - \phi_{0,i}^{S} + N_{r,i}^{S}\right) + \lambda_i \varepsilon_\phi \\ &= \rho_r^S + c\left(dt_r(t_r) - dT^S(t^S)\right) - I_{r,i}^{S} + T_r^S + \lambda_i B_{r,i}^{S} + d\Phi_{r,i}^{S} + \varepsilon_\phi \end{aligned} \tag{2.5}$$

นำสมการที่ 2.6 และ 2.7 ไปแทนค่าตัวแปรในสมการที่ 2.4 และ 2.5

$$B_{r,i}^{S} = \phi_{r,0,i} - \phi_{0,i}^{S} + N_{r,i}^{S} \tag{2.6}$$

$$\begin{aligned} d\Phi_{r,i}^{S} &= -d_{r,pco,i}^{T} e_{r,enu}^{S} + \left(E^S d_{pco,i}^{S}\right)^T e_r^S \\ &\quad + d_{r,pcv,i}(El) + d_{r,pcv,i}^{S}(\theta) - d_{r,disp}^{T} e_{r,enu}^{S} + \lambda_i \phi_{pw} \end{aligned} \tag{2.7}$$

โดยที่

$\phi_{r,j}^{S}$ คือเฟสของสัญญาณคลื่นพาห์จากดาวเทียม (นับเป็นลูกคลื่น) ย่านความถี่ L_i

$\phi_{r,i}(t)$ คือจำนวนเฟสของวงจรกำเนิดสัญญาณตัวรับท้องถิ่นย่านความถี่ L_i ณ เวลา t

$\phi_i^s(t)$ คือจำนวนเฟสของสัญญาณนำร่องย่านความถี่ L_i ณ เวลา t

$\phi_{r,0,i}$ คือจำนวนเฟสเริ่มต้นของวงจรกำเนิดสัญญาณตัวรับท้องถิ่นย่านความถี่ L_i

$\phi^{s}_{r,0,i}$ คือจำนวนเฟสเริ่มต้นของสัญญาณนำร่องย่านความถี่ L_i ที่ถูกส่ง ณ เวลา t_0

คือความยาวเฟส ย่านความถี่ L_i ในหน่วยเมตร

λ_i คือความยาวคลื่น ย่านความถี่ L_i ในหน่วยเมตร

$B^{s}_{r,i}$ คือจุดทำงานของเฟสคลื่นพาห์ย่านความถี่ L_i

$\Phi^{s}_{r,i}$ คือค่าแก้เฟสคลื่นพาห์ย่านความถี่ L_i

$N^{s}_{r,i}$ คือค่าเริ่มต้นในการวัดเฟสสัญญาณ

ρ^{s}_{n} คือระยะเรขาคณิตระหว่างดาวเทียมและสายอากาศตัวรับ

$dt_r(t_r)$ คือจุดทำงานของสัญญาณนาฬิกาของตัวรับ ณ เวลาที่รับสัญญาณ

$d_sT^s(t^s)$ คือจุดทำงานของสัญญาณนาฬิกาของดาวเทียม ณ เวลาที่ส่งสัญญาณ

$I^{s}_{r,i}$ คือเวลาหน่วงจากชั้นบรรยากาศไอโอโนสเฟียร์

T^{s}_{r} คือเวลาหน่วงจากชั้นบรรยากาศโทรโปสเฟียร์

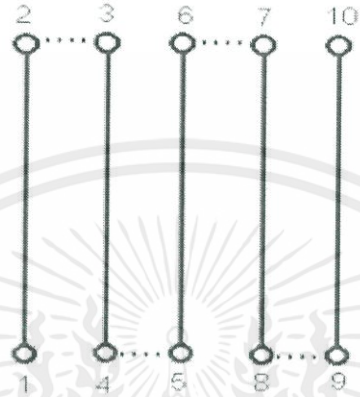
ε_{φ} คือค่าความผิดพลาดในการวัดเฟส

จากสมการที่ 2.4 จะทำให้เราทราบจำนวนเฟสสัญญาณคลื่นพาห์ซึ่งสามารถนำไปคำนวณหาระยะเทียมระหว่างดาวเทียมถึงตัวรับสัญญาณบนโลกได้ โดยนำไปคูณกับค่าความยาวคลื่นของสัญญาณคลื่นพาห์ตามสมการที่ 2.5 ในทางปฏิบัติ การหาพิกัดของอุปกรณ์ตัวรับจะต้องต่อสายอากาศเข้ากับตัวรับสัญญาณ และขณะที่รับสัญญาณนั้น โลกก็มีการเคลื่อนที่ตลอดเวลาด้วย ดังนั้นการหาระบุตำแหน่งของตัวรับให้มีความถูกต้องมากที่สุด จะต้องรวมจุดการทำงานของเฟสคลื่นพาห์และค่าแก้เฟสสัญญาณคลื่นพาห์เข้าไปด้วย โดยในสมการที่ 2.6 จะมีตัวแปร $B^{s}_{r,i}$ ซึ่งเป็นค่าจุดทำงานของเฟสคลื่นพาห์ย่านความถี่ L_i และสมการที่ 2.7 $\Phi^{s}_{r,i}$ เป็นค่าแก้เฟสสัญญาณคลื่นพาห์ย่านความถี่ L_i รวมอยู่ในสมการที่ 2.5 แล้ว

2.5 ส่วนการสร้างเส้นทางการเดินรถทางเกษตรกรรม

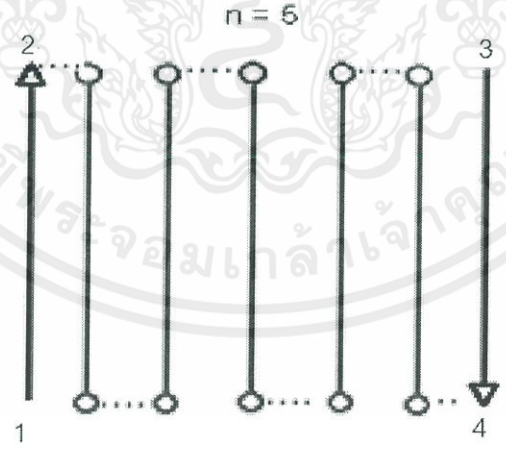
ในการสร้างเส้นทางการเดินรถทางเกษตรกรรม จำเป็นต้องมีค่าพิกัดตำแหน่งมาเก็บไว้ก่อน โดยในปริญญานิพนธ์นี้จะมีวิธีในการเก็บพิกัดตำแหน่ง 2 ชนิด คือแบบเก็บพิกัดทีละจุดโดยเก็บ

จากพิกัดที่แสดงบนส่วนต่อประสานกราฟิกกับผู้ใช้ และเก็บพิกัดที่มุมไร่ทางเกษตรกรรม 4 จุด เพื่อสร้างเส้นตรงที่ขนานกัน ดังรูปที่ 2.6 และ 2.7



รูปที่ 2.6 การเก็บพิกัดตำแหน่งเส้นทางทีละจุด

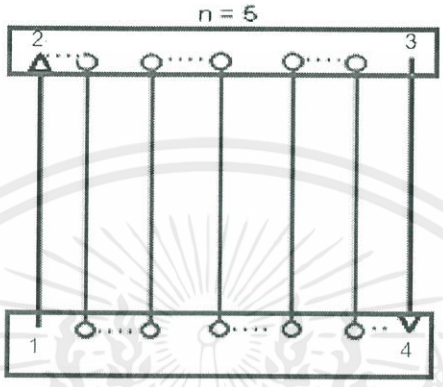
ในรูปที่ 2.6 ตัวเลข 1,2,..,10 แสดงถึงการเก็บพิกัดในแต่ละจุด หากต้องการเก็บพิกัดเส้นทาง 10 จุด ก็จะต้องเก็บพิกัด 10 ครั้ง และหากพิกัดตำแหน่งมีความเป็นเส้นขนานกัน อาจเก็บพิกัดตำแหน่งได้โดยพิกัดที่มุมไร่ทางเกษตรกรรม 4 จุด ดังรูป 2.7



รูปที่ 2.7 การเก็บพิกัดตำแหน่งเส้นทางที่มุมไร่ทางเกษตรกรรม 4 จุด

ในรูปที่ 2.7 จะทำการเก็บพิกัดเส้นทางที่มุมไร่ทางเกษตรกรรม 4 จุด คือจุด 1,2,3,4 ตามลำดับ และทำการป้อนระยะความห่างของแต่ละแนวเส้น

ในการคำนวณพิกัดของเส้นขนานนั้น ทำได้โดยการแปลงพิกัดละติจูด และลองจิจูดเป็นพิกัดแบบ utm แล้วคำนวณระยะระหว่างจุดที่ 1 กับ 4 และ 2 และ 3 ดังรูปที่ 2.8

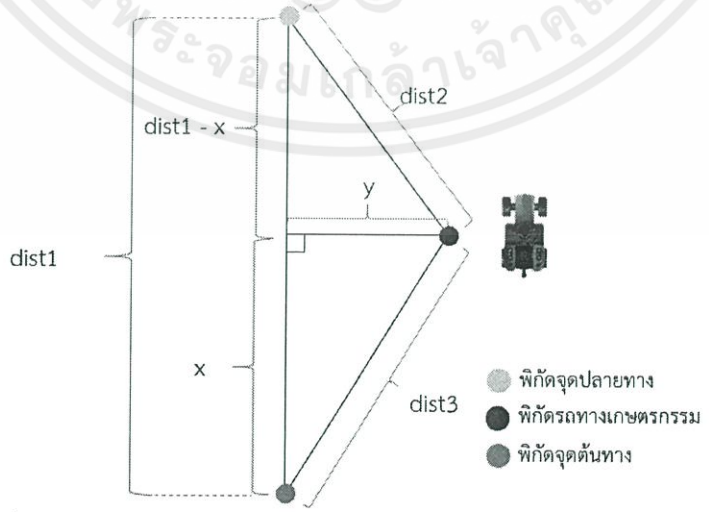


รูปที่ 2.8 การหาระยะของจุดพิกัดที่มุมไร่ทางเกษตรกรรม

ในรูปที่ 2.8 หลังจากที่หาระยะในหน่วยเมตรของจุดที่ 1 กับ 4 และ 2 กับ 3 พร้อมทั้งหาทิศทางในแนวแกน utm แล้วจะทำการแบ่งระยะนั้นเป็นส่วนๆ ตามจำนวนเส้นที่ต้องการ แล้วนำระยะนั้นพร้อมกับมุม มาคำนวณกลับหาเส้นขนานที่ต้องการ

2.6 การหาระยะทางที่สั้นที่สุดระหว่างรถทางเกษตรกรรม และเส้นทาง

ในกรณีที่มีเส้นทางเดินรถเป็นเส้นตรง และมีจุดพิกัดอ้างอิงเป็นจุดพิกัดต้นทาง และจุดพิกัดปลายทาง จะสามารถหาระยะห่างระหว่างรถทางเกษตรกรรม และเส้นทางได้ ดังรูปที่ 2.9



รูปที่ 2.9 การหาระยะห่างระหวางรถทางเกษตรกรรมกับเส้นทางที่กำหนด

จากรูปที่ 2.9 แสดงถึงตำแหน่งอ้างอิง และระยะห่างระหว่างตำแหน่งของรถทางเกษรกรรม และตำแหน่งอ้างอิง สำหรับใช้ในการคำนวณระยะทางที่สั้นที่สุดระหว่างรถทางเกษตรกรรม และเส้นทาง โดยที่

dist1 คือ ระยะระหว่างพิกัดต้นทาง และปลายทาง

dist2 คือ ระยะระหว่างรถทางเกษตรกรรมกับปลายทาง

dist3 คือ ระยะระหว่างรถทางเกษตรกรรมกับต้นทาง

y คือ ระยะที่ใกล้ที่สุดระหว่างรถทางเกษตรกรรมกับเส้นทาง

x คือ ระยะที่ใช้ในการคำนวณหา y และหาค่ามุมของรถทางเกษตรกรรมกับเส้นทาง

ทำการหาระยะทางระหว่างพิกัดที่อยู่ในรูปของละติจูด และลองจิจูดโดยอาศัย Haversine formula ซึ่งเป็นระยะห่างที่คำนึงถึงความโค้งของผิวโลก ดังสมการที่ 2.8

$$d = 2r\,arcsin(\sqrt{sin^2(\frac{\varphi_2-\varphi_1}{2}) + cos(\varphi_1)\,cos(\varphi_2)\,sin^2(\frac{\lambda_2-\lambda_1}{2})}) \quad (2.8)$$

โดยที่

r คือ รัศมีของโลก

φ_2, φ_1 คือ ละติจูดของจุดต้นทาง และละติจูดของจุดปลายทาง

λ_2, λ_1 คือ ลองจิจูดของจุดต้นทาง และลองจิจูดของจุดปลายทาง

เมื่อได้ระยะห่างจากการคำนวนโดยสมการที่ 2.8 จากนั้นจะใช้ทฤษฎีบทพีทาโกรัส ในการหาระยะห่างระหว่างรถทางเกษตรกรรม และเส้นทาง ดังสมการที่ 2.9 และ 2.10

$$(dist2)^2 = (dist1 - x)^2 + y^2 \quad (2.9)$$

$$(dist3)^2 = x^2 + y^2 \quad (2.10)$$

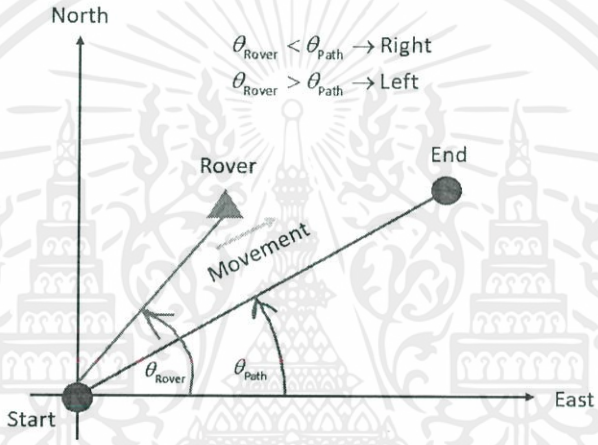
เมื่อได้ผลการคำนวณจากสมการที่ 2.9 และ 2.10 จะสามารถหาระยะห่างที่ใกล้ที่สุดระหว่างรถทางเกษตรกรรม และเส้นทางได้ ดังสมการที่ 2.11

$$y = ((dist2)^2 - (\frac{(dist1)^2-(dist2)^2+(dist3)^2}{2dist1}))^{0.5} \quad (2.11)$$

จากค่า y ที่ได้ เมื่อนำไปใช้ร่วมกับเซนเซอร์ GY-85 จะสามารถใช้ข้อมูลดังกล่าวเพื่อแก้ไขเส้นทางการเดินรถได้อย่างแม่นยำและถูกต้องมากขึ้น

2.7 การหาทิศทางระหว่างรถทางเกษตรกรรม และเส้นทาง

ในกรณีที่มีเส้นทางเดินรถเป็นเส้นตรง และมีจุดพิกัดอ้างอิงเป็นจุดพิกัดต้นทาง และจุดพิกัดปลายทาง จะสามารถหาทิศทางของรถทางเกษตรกรรม และเส้นทางได้ โดยนำพิกัดของเส้นทาง และตัวรถทางเกษตรกรรม มาแสดงในรูปของพิกัดสองมิติแล้วเปรียบเทียบกับแกนทิศตะวันออก ดังรูปที่ 2.10



รูปที่ 2.10 การหาระยะห่างระหว่างรถทางเกษตรกรรมกับเส้นทางที่กำหนด

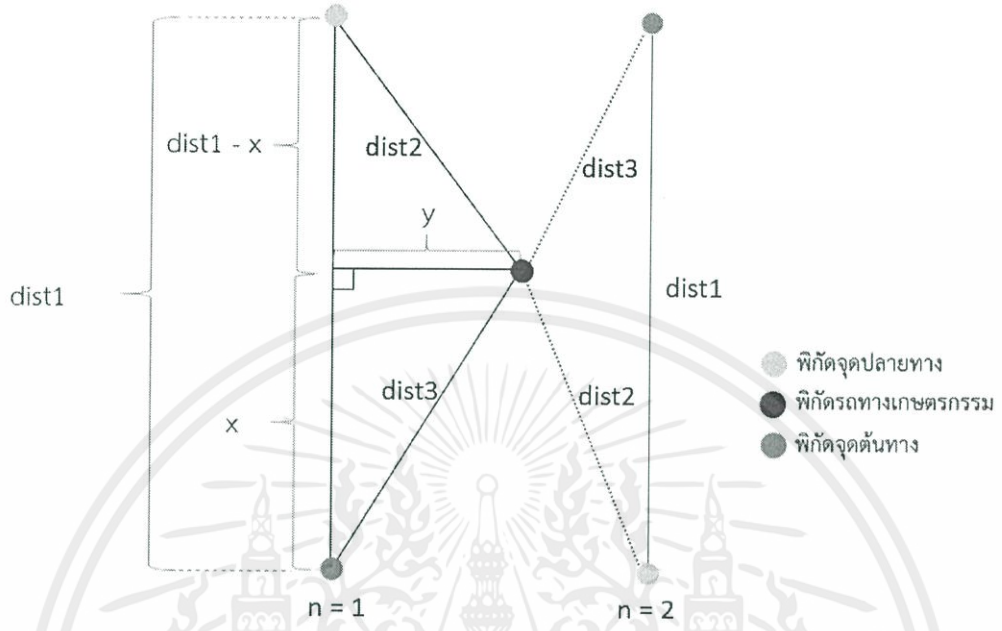
จากรูปที่ 2.10 เราสามารถนำพิกัดของเส้นทางมาแสดงทิศทาง ให้ผลลัพธ์ออกมาเป็นมุมที่กระทำกับแกนทิศตะวันออก โดยมีจุดพิกัดต้นทางเป็นจุดอ้างอิง (0,0) และนำมาเปรียบเทียบกับมุมของจุดพิกัดตำแหน่งปลายทาง และจุดพิกัดของรถทางเกษตรกรรม จะได้ผลลัพธ์สองรูปแบบ คือ

มุมที่จุดพิกัดของรถทางเกษตรกรรมทำกับมุมของจุดพิกัดปลายทาง เมื่อเปรียบเทียบกับมุมของจุดพิกัดต้นทางมีค่าน้อยกว่า จะได้ว่ารถทางเกษตรกรรมอยู่ทางขวาของเส้นทาง

มุมที่จุดพิกัดของรถทางเกษตรกรรมทำกับมุมของจุดพิกัดปลายทาง เมื่อเปรียบเทียบกับมุมของจุดพิกัดต้นทางมีค่ามากกว่า จะได้ว่ารถทางเกษตรกรรมอยู่ทางซ้ายของเส้นทาง

2.8 การเลือกเส้นทางที่ใกล้เคียงที่สุด เพื่อระยะห่างกับรถทางเกษตรกรรม

ในกรณีที่มีเส้นทางเดินรถเป็นเส้นตรง และมีจุดพิกัดอ้างอิงเป็นจุดพิกัดต้นทาง และจุดพิกัดปลายทางหลายจุด ดังรูปที่ 2.11



รูปที่ 2.11 เส้นทางสองเส้น (n = 1 และ n = 2)

จากรูปที่ 2.11 ในกรณีที่รถทางเกษตรกรรมอยู่ระหว่างเส้นทางหลายเส้น จะต้องทำการเลือกเส้นทางที่ใกล้กับรถทางเกษตรกรรมที่สุด โดยสามารถคำนวณหาค่า index เพื่อให้ความสำคัญกับเส้นทางแต่ละเส้น โดยเส้นที่มี index ต่ำที่สุดจะถูกใช้ในกาคำนวณระยะห่างระหว่างรถทางเกษตรกรรมกับเส้นทาง โดย

dist1 คือ ระยะระหว่างพิกัดต้นทาง และปลายทาง

dist2 คือ ระยะระหว่างรถทางเกษตรกรรมกับปลายทาง

dist3 คือ ระยะระหว่างรถทางเกษตรกรรมกับต้นทาง

y คือ ระยะที่ใกล้ที่สุดระหว่างรถทางเกษตรกรรมกับเส้นทาง

x คือ ระยะที่ใช้ในการคำนวณหา y และหาค่ามุมของรถทางเกษตรกรรมกับเส้นทาง

ทำการหาค่าความสำคัญของเส้นทาง โดยในแต่ละเส้นทางจะมีตัวแปร dist1,dist2,dist3 ที่แตกต่างกัน ทำให้การคำนวณเส้นทางที่อยู่ใกล้จะมีค่า index ต่ำกว่าเส้นทางที่อยู่ไกล และเส้นทางที่ยาวจะมีค่า index ที่ต่ำกว่าเส้นทางที่สั้น ดังสมการที่ 2.12

$$s = \frac{dist1+dist2+dist3}{2} \tag{2.12}$$

$$index = \sqrt{s|s-dist1||s-dist2||s-dist3|} \tag{2.13}$$

จากค่า index ที่ได้ หากเส้นทางใดที่มีค่า index ต่ำที่สุด จะนำมาใช้เป็นเส้นทางที่นำมาคำนวณหาระยะห่างและทิศของรถทางเกษตรกรรม 1 เส้นทาง จากนั้นจะแสดงผลผ่านส่วนต่อประสานกราฟฟิกกับผู้ใช้ต่อไป

2.9 โมดูล Raspberry Pi 3

โมดูล Raspberry Pi 3 คือ บอร์ดประมวลผลขนาดเล็กที่มีรูปแบบการทำงานคล้ายกับคอมพิวเตอร์เครื่องหนึ่ง สามารถรับข้อมูลจากคีย์บอร์ด จอมอนิเตอร์ และเมาส์ได้ นอกจากนี้ยังมีระบบการเชื่อมต่ออินเตอร์เน็ตทั้งแบบใช้สายและไร้สาย มีประสิทธิภาพในการทำงานได้หลากหลาย ไม่ว่าจะเป็นการเล่นเกม การใช้งานอินเตอร์เน็ตสำหรับการส่งอีเมล หรือการเล่นไฟล์วีดีโอที่มีความละเอียดสูง (High-Definition) รองรับการติดตั้งระบบปฏิบัติการของ Linux (Linux Operating System) ได้หลายระบบ เช่น Arch Linux Pidora (Fedora) Raspbian (Debian) เป็นต้น ซึ่งติดตั้งระบบบน SD Card

Raspberry Pi ได้ถูกออกแบบให้ภายในบอร์ดมีหน่วยประมวลผลหลายชนิด เช่น CPU GPU RAM และมีจุดเชื่อมต่อ GPIO ให้สามารถเชื่อมต่อกับอุปกรณ์อื่นได้ ซึ่งมีรายละเอียดและโครงสร้าง แสดงดังต่อไปนี้

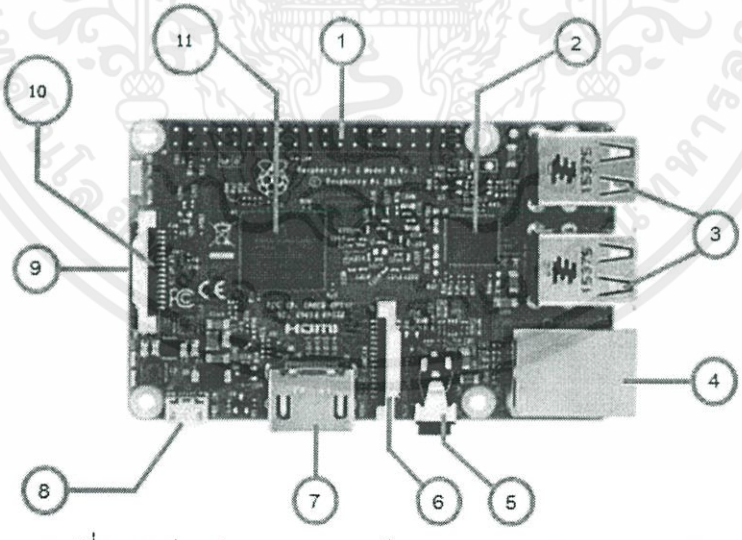
ตารางที่ 2.2 รายละเอียดของ Raspberry Pi 3

Features	Raspberry Pi 3		
	Model A	Model B	Model B+
CPU	ARM1176JZF-S 700 MHz		
GPU	Broadcom Video Core IV 250 MHz		
Memory	256 MB SDRAM	512 MB SDRAM	
Flash Memory	-		
Memory Card support	SD, MMC, SDIO		Micro-SD
USB	1 Port	2 Ports	4 Ports
Ethernet	-	10/100 Mb/s	
WIFI	-		

ตารางที่ 2.2 รายละเอียดของ Raspberry Pi 3 (ต่อ)

Peripherals	8xGPIO, UART, I2C bus, I2S audio, SPI, HDMI, Composite RCA, 3.5 mm jack	17xGPIO, HDMI, Composite RCA, 3.5 mm jack
Operating Voltage	5 V	
Operating System	Linux	
Size	3.37''x2.13''	
Weight	45 g	

จากตารางที่ 2.2 แสดงให้เห็นถึงความแตกต่างของ Raspberry Pi โมเดลต่างๆ โดยโครงงานนี้เลือกใช้งาน Raspberry Pi 3 Model B เนื่องจากมีความเหมาะสมต่อการใช้งานมากที่สุด ทั้งในด้านของระบบประมวลผล และราคาของอุปกรณ์ โดยสามารถเชื่อมต่อการทำงานกับเครื่องรับจีเอ็นเอสเอสแบบหนึ่งความถี่ได้ ผ่าน Port USB นอกจากนี้ยังมีระบบ I2C เพื่อทำการเชื่อมต่ออุปกรณ์เซ็นเซอร์ ตลอดจนมี Port HDMI สำหรับเชื่อมต่อไปยังจอแสดงผล ซึ่งคุณสมบัติที่กล่าวมานี้เพียงพอต่อการใช้งานสำหรับการออกแบบ และพัฒนาระบบนำร่องทางเกษตรกรรม



รูปที่ 2.12 ส่วนประกอบของบอร์ด Raspberry Pi 3 model B

หมายเลข 1 จุดเชื่อมต่อ GPIO มีคุณสมบัติของ PIN แสดงดังรูปที่ 2.13

หมายเลข 2 ชิปควบคุม LAN (LAN Controller)

หมายเลข 3 พอร์ต USB 2.0 จำนวน 4 พอร์ต

หมายเลข 4 พอร์ต RJ-45 Ethernet LAN 10/100Mbps

หมายเลข 5 จุดเชื่อมต่อสัญญาณเสียงขนาด 3.5 มิลลิเมตร

หมายเลข 6 จุดเชื่อมต่อกล้องแบบ CSI

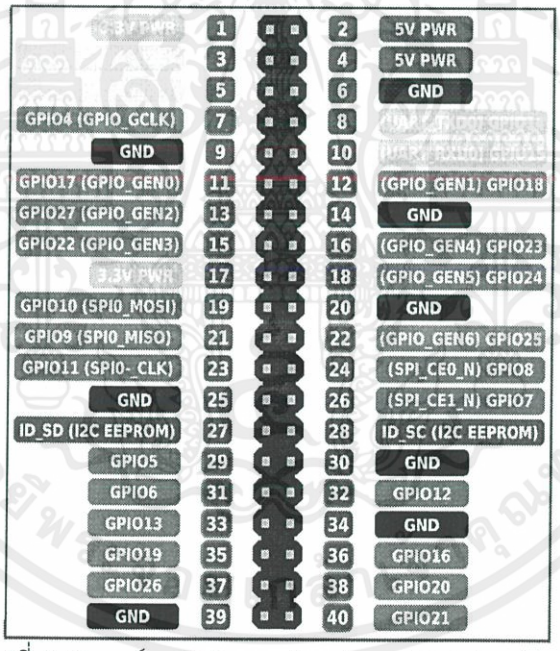
หมายเลข 7 พอร์ตเชื่อมต่อภาพและเสียงแบบ HDMI

หมายเลข 8 พอร์ต Micro USB Power สำหรับจ่ายกระแสไฟให้กับวงจรบอร์ด Raspberry Pi 3

หมายเลข 9 ช่องใส่ไมโคร SD Card (ด้านหลัง)

หมายเลข 10 พอร์ต DSI (Display Serial Interface) ใช้สำหรับต่อจอแสดงผล เช่น จอแสดงผลแบบ TFT Touch Screen เป็นต้น

หมายเลข 11 ชิป Broadcom VideoCore IV 250 MHz



รูปที่ 2.13 พอร์ต GPIO ของ Raspberry Pi 3 Model B

โปรแกรมที่ใช้ควบคุมการทำงานภายในของ Raspberry Pi 3 จะใช้ภาษาไพธอน (Python) เนื่องด้วยเหตุผลดังต่อไปนี้

1. สามารถทำงานได้หลายระบบปฏิบัติการ หากผู้เขียนโปรแกรมเขียนจากแพลตฟอร์มใดๆ แล้วสามารถนำโปรแกรมที่ได้ไปใช้งานต่างแพลตฟอร์มที่เขียนได้

2. ไม่ต้องเสียค่าใช้จ่ายในการจัดซื้อโปรแกรมต้นฉบับ โดยปกติแล้วโปรแกรมภาษาทั่วไปจะต้องจัดซื้อโปรแกรมต้นฉบับเพื่อนำมาติดตั้งในราคาที่แพงมาก แต่โปรแกรมภาษาไพธอนสามารถ

ดาวน์โหลดจาก www.python.org ได้โดยตรง แล้วนำมาติดตั้งและศึกษาการใช้ด้วยตนเอง เพราะเป็นโปรแกรมประเภท Open Source

3. ภาษาไพธอนได้นำเอาข้อดีของแต่ละโปรแกรมภาษามาใช้ เช่น ภาษา C ภาษา C++ ภาษา Java และภาษา Perl โดยข้อดีของแต่ละภาษามีดังนี้

- ภาษา C เป็นภาษาที่สามารถใช้ในไมโครคอมพิวเตอร์ มินิคอมพิวเตอร์ หรือคอมพิวเตอร์ระดับเมนเฟรม

- ภาษา C++ เป็นภาษาที่สามารถเขียนโปรแกรมภาษา C ได้ทั้งหมด ซึ่งใช้งานง่ายกว่าภาษา C

- ภาษา Java เป็นภาษาที่มีการตรวจสอบข้อผิดพลาด ในตอน compile และ runtime ข้อผิดพลาดที่อาจเกิดขึ้นในโปรแกรมจึงน้อย และสามารถแก้จุดบกพร่องของโปรแกรมได้ง่าย

- ภาษา Perl เป็นภาษาที่ง่ายต่อการเรียนรู้และใช้งาน เพราะมีความยืดหยุ่นสูงจึงเหมาะงานด้านข้อความ และ Text File

4. มีความปลอดภัยสูง เนื่องจากภาษาไพธอนจะใช้งานในด้าน Server เป็นหลัก ซึ่งผู้ใช้งานทั่วไปไม่สามารถเข้าถึง Server ได้โดยตรงจึงมีความปลอดภัยสูงกว่า

5. ใช้ในการพัฒนา Web Service ซึ่งในปัจจุบันการพัฒนาซอฟต์แวร์ได้เน้นที่มีการแลกเปลี่ยนข้อมูลซึ่งกันและกันทั้งในองค์กรเดียวกัน หรือแม้แต่ต่างองค์กรกัน ทำให้เกิดความสะดวกสบาย ไม่ต้องใช้ซอฟต์แวร์อื่นๆ มาแปลงข้อมูลเพื่อให้เข้ากันได้อีกต่อไป และเรียนรู้ได้เร็วกว่าโปรแกรมภาษาอื่นๆ เพราะมีโครงสร้างภาษาที่ไม่ซับซ้อน ซึ่งโครงสร้างภาษาคล้ายคลึงกับภาษา C ถ้าโปรแกรมเมอร์ที่เคยใช้ภาษา C มาก่อนจะทำให้เรียนรู้ได้เร็วยิ่งขึ้น นอกจากนี้การเขียนโปรแกรมด้วยภาษาไพธอนจะมีความกระชับ และสั้นกว่าภาษา C

2.10 หลักการของเซนเซอร์ GY-85

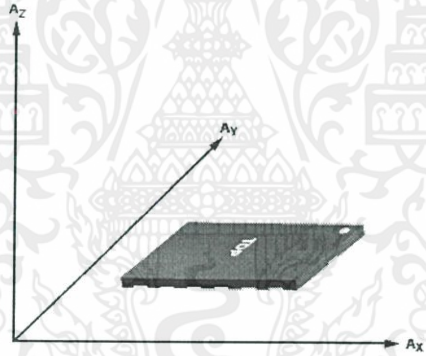
IMU (Inertial Measurement Unit) ประกอบไปด้วยเซ็นเซอร์ 3 ชนิด ได้แก่เซนเซอร์วัดความเร่ง (Accelerometer) เซนเซอร์วัดการหมุน (Gyroscope) และเซนเซอร์วัดความเข้มสนามแม่เหล็ก (Magnetometer) โดยเมื่อนำค่าที่เซนเซอร์ทั้งสสามชนิดนี้ตรวจวัดได้มาประมวลผลจะแสดงถึงการเคลื่อนที่ของวัตถุบนแกนอ้างอิงทั้งสาม คือแกน X Y และ Z

2.10.1 หลักการทำงานของเครื่องวัดความเร่ง (Accelerometer)

เครื่องวัดความเร่งภายในเซนเซอร์ GY-85 เป็นรุ่น ADXL345 มีลักษณะดังแสดงในรูปที่ 2.14 ใช้สำหรับวัดความเร่งของการเคลื่อนที่ของวัตถุในแนวแกน X Y และแกน Z

รูปที่ 2.14 เซนเซอร์ ADXL345

จากรูปที่ 2.14 ภายในเซนเซอร์ ADXL345 ประกอบด้วยสปริงขนาดเล็ก สำหรับวัดแรงที่กระทำต่อเซนเซอร์ และแปลงผลที่ได้ให้เป็นผลทางไฟฟ้า โดยอยู่ในรูปของแรงดันไฟฟ้าที่มีปริมาณแตกต่างกันตามค่าของแรงกระทำที่วัดได้ จากนั้นนำค่าที่ได้ไปคำนวณหาค่าความเร่งแกน X (Ax) แกน Y (Ay) และแกน Z (Az) โดยมีทิศทางการวางตัวของแต่ละแกนดังแสดงในรูปที่ 2.15

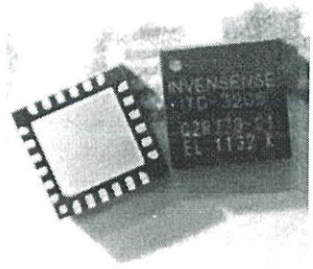


รูปที่ 2.15 แนวการวางตัวของแกน X Y และแกน Z ของเซนเซอร์ ADXL345

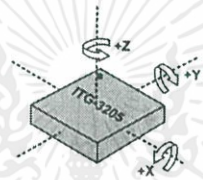
จากรูปที่ 2.15 สรุปได้ว่าค่าความเร่งที่เซนเซอร์ ADXL345 วัดได้ จะมีทิศทางตรงข้ามกับความเร่งหรือแรงที่มากระทำ โดยแรงที่มากระทำจะเป็นแรงโน้มถ่วงของโลก เมื่อนำค่าความเร่งที่วัดได้มาคำนวณด้วยหลักคณิตศาสตร์ จะสามารถหามุมที่เอียงไปของวัตุได้ ส่งผลให้ทราบแนวการวางตัวของวัตถุซึ่งเปรียบเทียบกับแนวแกน X Y และ Z

2.10.2 หลักการทำงานของเครื่องวัดการหมุน (Gyroscope)

หน้าที่หลักของ Gyroscope คือวัดความเร็วเชิงมุมของแกน X แกน Y และแกน Z โดยภายในเซ็นเซอร์ GY-85 มีเครื่องวัดการหมุน คือ เซนเซอร์ ITG-3205 ซึ่งมีลักษณะดังรูปที่ 2.16 และมีแนวการวางตัวแกน X แกน Y และแกน Z ดังรูปที่ 2.17



รูปที่ 2.16 เซนเซอร์ ITG-3205

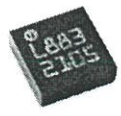


รูปที่ 2.17 แนวการวางตัวแกน X แกน Y และแกน Z ของเซนเซอร์ ITG-3205

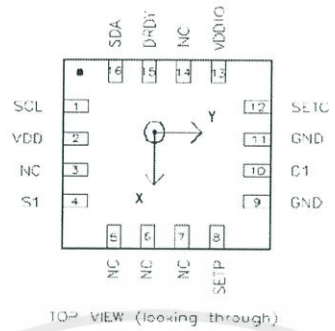
จากรูปที่ 2.16 และ 2.17 แสดงลักษณะเฉพาะของเซนเซอร์ ITG-3205 ในส่วนของลักษณะทางกายภาพ และแนวการวางตัวของแกนหลักทั้งสาม สำหรับการวัดความเร็วเชิงมุมของ ITG-3205 มีหลักการคล้ายคลึงกับการวัดความเร่งของแกนหลักทั้งสาม โดยภายใน Gyroscope จะมีสปริงอยู่ภายใน เมื่อเกิดการสั่นสะเทือนสปริงจะสร้างแรงดันไฟฟ้าขึ้น จากนั้นแรงดันฟ้านี้จะถูกนำไปประมวลผล และแสดงค่าความเร็วเชิงมุม ณ เวลาต่างๆออกมา

2.10.3 หลักการทำงานของเครื่องวัดความเข้มสนามแม่เหล็ก (Magnetometer)

หน้าที่หลักของ Magnetometer คือการตรวจจับความแรงของสนามแม่เหล็กในแกนต่างๆ ได้แก่ แกน X แกน Y และแกน Z เครื่องวัดความเร่งภายในเซนเซอร์ GY-85 เป็นรุ่น HMC5883L ดังแสดงในรูปที่ 2.18 โดยสามารถหาทิศทางการเคลื่อนที่ของวัตถุ ได้จากการเปรียบเทียบความแรงของสนามแม่เหล็กที่วัดได้ในแต่ละแกน ดังแสดงในรูป 2.19

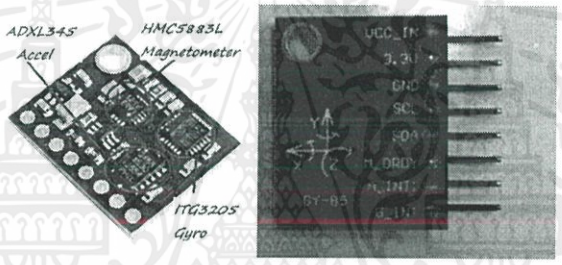


รูปที่ 2.18 เซนเซอร์ HMC5883L



รูปที่ 2.19 แนวการวางตัวแกน X แกน Y และแกน Z ของเซนเซอร์ HMC5883L

2.10.4 ข้อมูลเฉพาะของ GY-85



รูปที่ 2.20 เซนเซอร์ GY-85

จากรูปที่ 2.20 GY-85 ประกอบไปด้วย Accelerometer (ADXL345) Gyroscope (ITG3205) และ Compass (HMC5883L) ซึ่งสามารถซึ่งมีหลักการทำงานของแต่ละอย่าง ดังที่ได้อธิบายไปแล้วในหัวข้อที่ 2.7.1 ถึงหัวข้อที่ 2.7.3 ในการใช้เซ็นเซอร์ชนิดนี้ร่วมกับหน่วยประมวลผลจะต้องเชื่อมต่อผ่านผ่าน Bus I2C กับหน่วยประมวลผล เพื่อส่งข้อมูลให้หน่วยประมวลไปใช้งานต่อไป รายละเอียดคุณสมบัติและข้อมูลการต่อใช้งานขาของอุปกรณ์ GY-85 ที่รับและส่งข้อมูล แสดงดังตารางที่ 2.3 และ 2.4 ตามลำดับ

ตารางที่ 2.3 คุณสมบัติของอุปกรณ์ GY-85

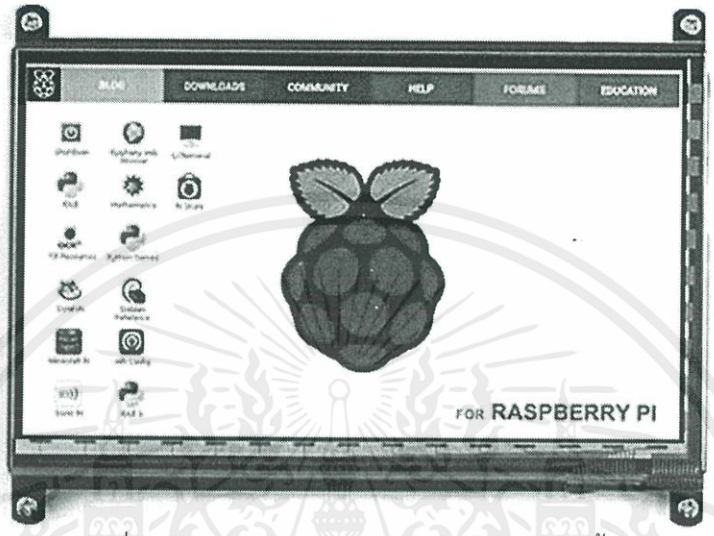
คุณสมบัติ	รายละเอียด
ไฟเลี้ยง	+3.3 ถึง +5 V
ชิป	ADXL345 HMC5883L ITG3205
การเชื่อมต่อ	บัส I2C

Accelerometer	ความละเอียดสูงสุด 16 เท่าของค่าความโน้มถ่วงโลก
Gyroscope	วัดการเปลี่ยนแปลงมุมได้สูงสุด 2000 องศาต่อวินาที
Compass	ความแม่นยำในการวัดทิศทางประมาณ 1 องศา

ตารางที่ 2.4 ข้อมูลการต่อใช้งานขาอุปกรณ์ GY-85

Pin	Name	Characteristic
1	VCC_IN	ขารับไฟ +5 โวลต์ไปที่ Regulate 3.3 โวลต์
2	3.3V	ขารับไฟ 3.3 โวลต์
3	GND	ขากราวนด์
4	SCL	ขาสัญญาณนาฬิกา บนบัส I2C
5	SDA	ขาสัญญาณข้อมูล บนบัส I2C
6	M_DRDY	Interrupt ของ HMC5883L
7	A_INT1	Interrupt ของ ADXL345B
8	G_INT	Interrupt ของ ITG3205

2.11 จอแสดงผล LCD แบบสัมผัสขนาด 7 นิ้ว



รูปที่ 2.21 จอแสดงผล LCD แบบสัมผัสขนาด 7 นิ้ว

จากรูปที่ 2.21 จอแสดงผลแบบ LCD แบบสัมผัสขนาด 7 นิ้ว เป็นอุปกรณ์การแสดงผลแบบจอสัมผัสที่รองรับการเชื่อมต่อกับอุปกรณ์ Raspberry Pi ผ่านทางพอร์ต HDMI และพอร์ต USB มีคุณสมบัติการใช้งาน คือ

1. เป็นหน้าจอแสดงผลแบบสัมผัสขนาด 7 นิ้ว (194 มม. x 110 มม. x 20 มม.) ที่มีความละเอียด 800 x 480 pixel

2. มีระบบมัลติทัช (Multi-Touch) สามารถรองรับจุดสัมผัสได้มากสุด 4 จุดสัมผัส

3. รองรับฟังก์ชั่นเสริมต่างๆ เช่น Backlight control mirror image เป็นต้น

จากคุณสมบัติข้างต้น จอแสดงผลแบบ LCD แบบสัมผัสขนาด 7 นิ้ว จึงมีความเหมาะสมในการใช้งาน โดยการรับคำสั่งการทำงานจากผู้ใช้ผ่านระบบสัมผัส และแสดงผลการทำงานผ่านส่วนต่อประสานกราฟิกกับผู้ใช้

2.12 หลักการมอเตอร์ไฟฟ้ากระแสตรง [12]

มอเตอร์ไฟฟ้า เป็นอุปกรณ์ไฟฟ้าชนิดหนึ่งที่เปลี่ยนจากพลังงานไฟฟ้าเป็นพลังงานกล ในปัจจุบันมอเตอร์แต่ละชนิด มีคุณสมบัติที่ต่างกันตามความเร็วรอบ หรือกำลังงานที่แตกต่างกัน

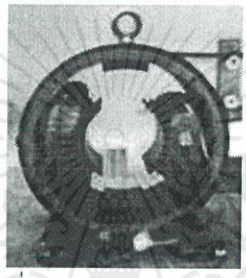
มอเตอร์ไฟฟ้ากระแสตรง เป็นกำลังขับเคลื่อนที่สำคัญอย่างหนึ่งในโรงงานอุตสาหกรรม เพราะสามารถปรับความเร็วได้ตามต้องการ พบมากในโรงงานอุตสาหกรรม เช่น โรงงานถลุงโลหะ โรงงานทอผ้า โรงงานเส้นใยโพลีเอสเตอร์ หรือใช้เป็นต้นกำลังในการขับเคลื่อนรถไฟฟ้า เป็นต้น

2.12.1 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงที่ส่วนประกอบที่สำคัญ 2 ส่วนดังนี้

2.12.1.1 สเตเตอร์ (Stator) เป็นส่วนที่อยู่กับที่ ประกอบด้วย

1) โครงของมอเตอร์ (Frame Or Yoke) เป็นโครงภายนอกที่ยึดส่วนประกอบต่างๆ ให้แข็งแรง อีกทั้งยังเป็นทางสำหรับเส้นแรงแม่เหล็กที่เคลื่อนจากขั้วเหนือไปยังขั้วใต้ ซึ่งทำด้วยเหล็กหล่อหรือเหล็กแผ่นหนาทรงกระบอก แสดงดังรูปที่ 2.22



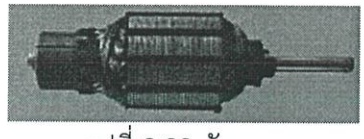
รูปที่ 2.22 โครงของมอเตอร์

2) ขั้วแม่เหล็ก (Pole) ประกอบด้วย 2 ส่วน คือ แกนขั้วแม่เหล็กและขดลวด

- ส่วนแรก คือ แกนขั้ว (Pole Core) ทำด้วยแผ่นเหล็กบางกลมซึ่งกั้นด้วยฉนวนประกอบกันเป็นแท่งยึดติดกับเฟรม ส่วนปลาย เป็นขั้วแม่เหล็ก (Pole Shoes) มีรูปร่างโค้งเพื่อรับรูปกลมของตัวหมุน ซึ่งขั้วแม่เหล็กและตัวหมุนต้องใกล้กันให้มากที่สุด เพื่อให้เกิดช่องว่างของอากาศน้อยที่สุดหรือไม่เกิดเลย ช่องว่างของอากาศนั้นมีผลต่อเส้นแรงแม่เหล็กจากขั้วแม่เหล็กไปยังตัวหมุนมีมากพอแล้วทำให้เกิดแรงบิดหรือกำลังบิดของตัวหมุนมากเป็นการทำให้มอเตอร์มีกำลังหมุน

- ส่วนที่สอง คือ ขดลวดสนามแม่เหล็ก (Field Coil) มีลักษณะพันอยู่รอบแกนขั้วแม่เหล็ก ขดลวดนี้จะทำหน้าที่รับกระแสจากภายนอกเพื่อสร้างเส้นแรงแม่เหล็ก โดยเส้นแรงแม่เหล็กนี้จะเกิดการหักล้างหรือเสริมกันกับสนามแม่เหล็กของอาเมเจอร์ ทำให้เกิดแรงบิดขึ้น

2.12.1.2 ตัวหมุน (Rotor) เป็นตัวที่ทำให้เกิดกำลังงาน โดยจะมีแกนวางอยู่ในตลับลูกปืน (Ball Bearing) ที่รวมอยู่ในแผ่นปิดหัวท้าย (End Plate) ของมอเตอร์ ประกอบด้วย 4 ส่วน ได้แก่



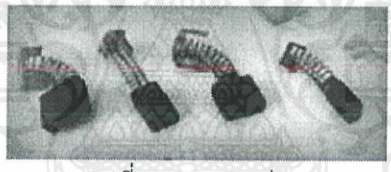
รูปที่ 2.23 ตัวหมุน

1) แกนเพลา (Shaft) เป็นตัวที่ไว้ยึดคอมมิวเตเตอร์ (Commutator) และแกนเหล็กอาร์มาเจอร์ (Armature Core) แกนเพลานี้จะวางบนแบริ่ง เพื่อบังคับให้หมุนโดยไม่เกิดการสั่นสะเทือน

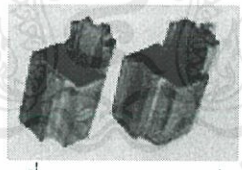
2) แกนเหล็กอาร์มาเจอร์ (Armature Core) สร้างจากแผ่นเหล็กลามิเนต (Laminated Sheet Steel) ซึ่งขดลวดอาร์เมเจอร์จะถูกพันที่บริเวณนี้

3) คอมมิวเตเตอร์ (Commutator) ทำมาจากแท่งทองแดงเล็กหลายแท่งที่รวมกันเป็นรูปทรงกระบอก แต่ละแท่งมีฉนวนไมก้า (mica) คั่นอยู่ ส่วนหัวของแท่งทองแดงในคอมมิวเตเตอร์ จะมีร่องไว้ใส่ปลายสายของขดลวดอาร์มาเจอร์ คอมมิวเตเตอร์ทำหน้าที่สัมผัสกับแปรงถ่าน (Carbon Brushes) เพื่อรับกระแสจากสายป้อนไปยังขดลวดอาร์มาเจอร์ เพื่อสร้างเส้นแรงแม่เหล็กที่จะนำไปหักล้างและเสริมกันกับเส้นแรงแม่เหล็กที่เกิดจากขดลวดขั้วแม่เหล็ก

4) ขดลวดอาร์มาเจอร์ (Armature Winding) เป็นขดลวดที่พันอยู่ในร่องสลอท (Slot) ของแกนอาร์มาเจอร์ ขนาดและจำนวนรอบของขดลวดขึ้นอยู่กับการออกแบบของตัวหมุนตามที่ต้องการนำไปใช้งาน



รูปที่ 2.24 แปรงถ่าน



รูปที่ 2.25 ซองแปรงถ่าน

แปรงถ่าน ดังรูปที่ 2.24 ทำมาจากคาร์บอนมีรูปร่างเป็นแท่งสี่เหลี่ยมผืนผ้า ภายในซองแปรงถ่าน ดังรูปที่ 2.25 จะมีสปริงกดอยู่เพื่อให้ถ่านนี้สัมผัสกับแท่งทองแดงในคอมมิวเตเตอร์ตลอดเวลา โดยกระแสจากภายนอกที่รับเข้ามาจะไปยังคอมมิวเตเตอร์เพื่อให้ขดลวดอาร์มาเจอร์เกิดแรงบิด ส่งผลให้มอเตอร์สามารถหมุนได้

2.12.2 หลักการของมอเตอร์กระแสไฟฟ้าตรง (Motor Action)

เมื่อมีกระแสไฟฟ้าตรงไปยังมอเตอร์ จะแบ่งไปยังคอมมิวเตเตอร์เพื่อให้ขดลวดอาร์มาเจอร์สร้างเส้นแรงแม่เหล็กขึ้น และไปยังขดลวดสนามแม่เหล็กเพื่อสร้างเส้นแรงแม่เหล็กขึ้น ทำให้เกิด

สนามแม่เหล็กขึ้น 2 สนาม ซึ่งคุณสมบัติของเส้นแรงแม่เหล็กจะไม่ตัดกัน หากทิศทางตรงกันข้ามจะหักล้างกัน ส่วนทิศทางเดียวกันจะเสริมกัน ทำให้เกิดแรงบิดภายในอาร์มาเจอร์บนแกนเพลาที่อยู่กับตลับลูกปืน ทำให้อาร์มาเจอร์สามารถหมุนได้ กลายมาเป็น ตัวหมุน (Rotor) ซึ่งเส้นแรงแม่เหล็กทั้งสองจะมีปฏิกิริยาต่อกัน ทำให้ตัวหมุนหมุนตามกฎมือซ้ายของเฟลมมิ่ง (Fleming'left hand rule)

2.13 หลักการแบตเตอรี่ [13]

เซลล์แบตเตอรี่ ประกอบด้วยแผ่นที่เป็นขั้วบวก แผ่นที่เป็นขั้วลบ และสารละลายที่เป็นของเหลว เรียกว่า อิเล็กทรอไลท์ เซลล์เหล่านี้อาจมีการปิดแผ่นสนิทหรือมีช่องให้สารละลายระเหยได้ ส่วนเซลล์ที่มีช่องระเหยได้จะใช้สารละลายเป็นของเหลวแผ่นที่เป็นขั้วบวก และแผ่นที่เป็นขั้วลบจะวางอยู่คู่กันในเซลล์ แบตเตอรี่ อาจจะมีแผ่นที่เป็นขั้วบวกและแผ่นที่เป็นขั้วลบหลายชุดวางขนานกันเพื่อให้ได้ไฟฟ้าที่จ่ายออกสูงขึ้น แผ่นที่เป็นขั้วบวกและแผ่นที่เป็นขั้วลบจะถูกป้องกันไม่ให้มีส่วนที่สัมผัสกันได้ แต่ไอออนสามารถถ่ายโอนจากแผ่นหนึ่งไปยังอีกแผ่นหนึ่งโดยผ่านสารละลายได้ ทำให้เกิดกระแสไฟฟ้าขึ้น ความต่างศักย์ทางไฟฟ้าระหว่างแผ่นที่เป็นขั้วบวกและแผ่นที่เป็นขั้วลบนี้ จะขึ้นอยู่กับปฏิกิริยาที่เกิดขึ้นระหว่างชนิดของสารที่ใช้ทำแผ่นที่เป็นขั้วบวก แผ่นที่เป็นขั้วลบ และชนิดของสารละลาย แต่ปริมาณกระแสไฟฟ้าที่ได้จากแบตเตอรี่ จะขึ้นอยู่กับพื้นที่ของแผ่นที่เป็นขั้วบวกและแผ่นที่เป็นขั้วลบ ระยะห่างระหว่างแผ่นที่เป็นขั้ว และความเข้มของสารละลาย ความจุของแบตเตอรี่มักวัดเป็นแอมแปร์ในเวลา 1 ชั่วโมง วิธีการวัดความจุได้มีการตั้งมาตรฐานโดยกำหนดเวลาคงที่ และวัดกระแสไฟฟ้าที่จ่ายออกมาในช่วงเวลาดังกล่าว ส่วนใหญ่มักกำหนดเป็นเวลานาน 8 ชั่วโมง ถ้าการกำหนดต่างกัน เช่น แบตเตอรี่ลูกหนึ่งจ่ายกระแสไฟฟ้า 20 แอมแปร์ ในเวลา 8 ชั่วโมงจะมีความจุ 160 แอมแปร์ต่อชั่วโมง ในเวลา 8 ชั่วโมง แต่ถ้าแบตเตอรี่ลูกนี้จ่ายกระแสไฟฟ้า 40 แอมแปร์ จะวัดความจุได้น้อยกว่า 160 แอมแปร์ต่อชั่วโมง แต่ในทางตรงกันข้ามถ้าให้แบตเตอรี่ลูกนี้จ่ายไฟฟ้าต่ำกว่า 20 แอมแปร์ ก็จะได้ค่าความจุมากกว่า 160 แอมแปร์ต่อชั่วโมง

2.13.1 ประเภทของแบตเตอรี่

วัสดุที่นำมาใช้ทำแผ่นที่เป็นขั้วบวก (แผ่น Anode) มีหลายชนิด เช่น ตะกั่ว แคดเมียม แมกนีเซียม และสังกะสี ซึ่งเป็นสารที่ปล่อยอิเล็กตรอนได้ง่าย ส่วนแผ่นที่เป็นขั้วลบ (แผ่น Cathode) อาจจะทำด้วยตะกั่วไดออกไซด์ นิกเกิล ปรอท และเงิน ซึ่งจะรับอิเล็กตรอนได้ง่าย เนื่องจากคุณสมบัติที่ได้จากการใช้วัสดุที่ต่างชนิดกันนั้น จึงสามารถแบ่งแบตเตอรี่ออกเป็น 2 ประเภท คือ

2.13.1.1 แบตเตอรี่ปฐมภูมิ

แบตเตอรี่ปฐมภูมิ เป็นแบตเตอรี่ทำที่มาจาก คาร์บอน ปรอท ลิเทียม และสังกะสี แบตเตอรี่ประเภทนี้ใช้งานได้ครั้งเดียวเมื่อใช้งานเสร็จแล้วต้องทิ้ง ไม่สามารถประจุไฟฟ้าได้อีก

ส่วนมากใช้งานกับเครื่องใช้ไฟฟ้าที่สามารถถือได้ มีราคาถูก และอายุการใช้งานน้อย เช่น ถ่านไฟฉาย แบตเตอรี่สำหรับวิทยุเล็ก เป็นต้น

2.13.1.2 แบตเตอรี่ทุติยภูมิ

แบตเตอรี่ทุติยภูมิ แบตเตอรี่ประเภทนี้สามารถประจุไฟกลับเข้าไปใหม่ได้ เมื่อไฟฟ้าหมดจึงสามารถนำมาทำให้ใช้งานได้อีก แบตเตอรี่ที่นิยมใช้กันมาก คือ ชนิดที่ทำจากตะกั่วกรด ซึ่งพบเห็นกันมากในงานด้านระบบการสื่อสาร โทรคมนาคม ไฟสำรอง ซึ่งถือแม้ว่าแบตเตอรี่ชนิดนี้จะมีอายุการทำงานยาวนานกว่า และมีสึกหรอน้อยกว่าแต่จะมีราคาแพงกว่าแบบปฐมภูมิหลายสิบเท่า

แบตเตอรี่แบบตะกั่วกับกรด ประกอบด้วยแผ่นที่เป็นขั้วบวกทำด้วยตะกั่วไดออกไซด์ (PbO_2) มีสีน้ำตาล แผ่นที่เป็นขั้วลบทำด้วยตะกั่วพรุน (Pb) มีสีเทามีสารละลายเป็นกรดกำมะถัน (H_2SO_4) ขณะที่จ่ายไฟฟ้าจะมีปฏิกิริยาเกิดขึ้นดังนี้

1) ออกซิเจน (O_2) จากตะกั่วไดออกไซด์ (PbO_2) จะรวมตัวกับไฮโดรเจน (H_2) ในกรดกำมะถัน

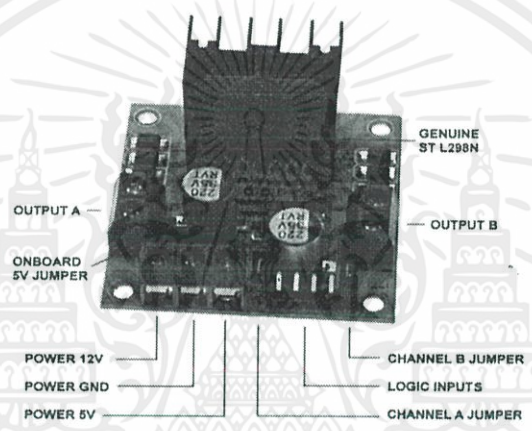
2) ตะกั่ว (Pb) จากตะกั่วไดออกไซด์ (PbO_2) จะรวมตัวกับอนุมูลซัลเฟตเป็น $(PbSO_4)$ ปฏิกิริยานี้เกิดขึ้นที่แผ่นที่เป็นขั้วบวกและลบจะไปเจือจางกรดกำมะถัน ดังนั้นเมื่อเกิดปฏิกิริยาไปนานๆ กระแสไฟฟ้าที่ได้จะลดลงเรื่อยๆ ขณะที่ประจุไฟฟ้าจะเกิดปฏิกิริยาตรงกันข้ามกับตอนจ่ายไฟฟ้า คือ ตะกั่วซัลเฟตจะแตกตัวเป็นตะกั่ว (Pb) กับอนุมูลซัลเฟต (SO_4^{2-}) ที่ทั้งสองแผ่นนั้นจะแตกตัวเป็นไฮโดรเจนกับออกซิเจน โดยที่ไฮโดรเจนจะรวมตัวกับอนุมูลซัลเฟตเป็น กรดกำมะถันและออกซิเจน และรวมตัวกับตะกั่วเป็นตะกั่วไดออกไซด์ (PbO_2)

2.13.2 การชาร์จแบตเตอรี่

แบตเตอรี่ที่มีการใช้งานทุกวันหรือเก็บไว้โดยไม่ได้ใช้งานเป็นระยะเวลานานๆ ก็จะทำให้กระแสไฟฟ้าในแบตเตอรี่หมดไป ดังเช่นแบตเตอรี่ที่อยู่กับเครื่องพ่นน้ำ ก็จะมีเครื่องประจุไปอยู่ใต้เครื่องพ่นน้ำด้วย สำหรับประจุไฟฟ้าชดเชยส่วนที่ใช้ไปเพื่อให้แบตเตอรี่มีไฟฟ้าเต็มก่อนการนำไปใช้งาน สำหรับแบตเตอรี่ที่ไม่ได้ใช้งานนานๆ กระแสไฟฟ้าในแบตเตอรี่ก็จะหมดไป ทำให้ต้องมีการชาร์จกระแสไฟฟ้าเข้าไปในแบตเตอรี่ใหม่ด้วยเครื่องประจุไฟฟ้า เพื่อให้มีกระแสไฟฟ้าใช้งานได้ทุกเวลา ในการชาร์จแบตเตอรี่เราต้องอาศัยวงจร Half Wave Rectifier และวงจร Filter การชาร์จแบตเตอรี่นั้นจากการผ่านกระบวนการดังกล่าวจะเกิดสัญญาณ (Sine Wave) แบบครึ่งคลื่นที่แรงดัน 13.5 ถึง 15 โวลต์ โดยมีหลักการใช้ความต่างศักดิ์ที่มากกว่าดึงประจุจากภายในออกมาทำให้มีความต่างศักย์เท่าเดิมและสามารถจ่ายกระแสไฟฟ้าได้เท่าเดิม

2.14 หลักการไอซี L298N [14]

IC L298N เป็นไอซีขับมอเตอร์แบบฟูลบริดจ์ สามารถขับมอเตอร์ได้ 2 ตัว ทำงานแบบแยกกัน สามารถควบคุมความเร็วและทิศทางมอเตอร์ด้วยการจ่าย Logic ให้หมุนไปด้านหน้า หมุนไปด้านหลังได้ แบบอิสระแยกกัน ใช้การควบคุมแบบ Full Speed หรือแบบพัลส์ สำหรับปรับ Speed ได้เช่นกัน ขับแรงดันโหลดได้สูงสุดถึง 46 โวลต์ ขับกระแสได้ช่องละ 2A และใช้ไฟ 4.5 ถึง 7 โวลต์ แต่เมื่อทดสอบจริงมีแรงดันไฟฟ้าขนาด 3.3 ถึง 5 โวลต์ สามารถเป็น Logic ได้



รูปที่ 2.26 IC L289

2.14.1 ขาของ IC L289

- ขา 1 ปกติต่อกับ Ground
- ขา 2 และ 3 ต่อเข้าขั้วมอเตอร์
- ขา 4 เป็นแรงดันสำหรับมอเตอร์ โดยคำนึงถึงการลักษณะมอเตอร์(Motor Specification) เป็นหลัก
- ขา 5 และ 7 เป็นขาสำหรับรับ Logic 1 (3.3 Volt+) Logic 0 (0 Volt) เพื่อควบคุมมอเตอร์ A
- ขา 6 Enable A เสมือนการเปิด-ปิด มอเตอร์ ถ้าจ่ายแต่ Logic เข้าขา 5 ขา 7 แต่ไม่จ่าย Enable A มอเตอร์ก็จะไม่ทำงาน เมื่อจ่ายแรงดัน 3.3 โวลต์ มอเตอร์จะทำงาน
- ขา 8 Ground ต้องเป็น Ground ร่วมขา 1 และ 15 ต้องต่อร่วมด้วยขา 9 แรงดันสำหรับ Logic
- ขา 10, 12 เป็นขาสำหรับรับ Logic 1 (3.3 Volt+) Logic 0 (0 Volt) เพื่อควบคุมมอเตอร์ B

- ขา 13 และ 14 ต่อเข้าขั้วมอเตอร์
- ขา 15 ต่อ Ground

โดยขา Enable สามารถจ่ายสัญญาณ PWM เพื่อควบคุมความเร็วมอเตอร์ได้ด้วย แต่ถ้าจะใช้มอเตอร์แบบเต็มสปีดก็สามารถจ่ายเป็น Logic 1 Logic 0 ได้ทันที และขา Input 3 และ 4 เมื่อจ่าย Logic 1 ที่ขา Input 3 และจ่าย 0 ที่ขา Input 4 มอเตอร์ก็จะหมุนไปทิศทางหนึ่ง แต่ถ้าจ่าย Logic กลับกันมอเตอร์ก็จะกลับทิศจากเดิม

โดยที่

- ENA: Enable A
- ENB: Enable B
- IN1 และ IN2: ควบคุมมอเตอร์ A
- IN3 และ IN4: ควบคุมมอเตอร์ B

2.15 ระบบฐานข้อมูล [15]

ฐานข้อมูล (Database) เป็นการรวบรวมข้อมูลจำนวนมาก และเก็บข้อมูลไว้อย่างเป็นระบบ โดยแยกเก็บตามรูปแบบความสัมพันธ์ของข้อมูล อาจมีการแยกเก็บในแฟ้มข้อมูลที่ต่างกันหรือเก็บไว้ในแฟ้มข้อมูลเดียวกันก็ได้

ระบบฐานข้อมูล (Database System) คือ ระบบที่ทำการเก็บรวบรวมข้อมูลที่มีความเกี่ยวข้องกันไว้ภายในแฟ้มข้อมูลเดียวกัน รวมถึงเปิดโอกาสให้ผู้ใช้งานสามารถประมวลผลข้อมูล และป้องกันข้อมูลภายในแฟ้มข้อมูลได้อย่างมีประสิทธิภาพ โดยมีระบบปฏิบัติการที่ใช้เป็นสื่อกลางระหว่างผู้ใช้กับโปรแกรมที่เกี่ยวข้องกับฐานข้อมูล เรียกว่า ระบบจัดการฐานข้อมูล (Database Management System : DBMS) ซึ่งช่วยให้ผู้ใช้งานสามารถสร้าง หรือแก้ไขข้อมูลภายในฐานข้อมูลได้อย่างมีประสิทธิภาพ

2.16 HTML [16]

HTML ถูกสร้างขึ้นในปี 1989 โดยนักฟิสิกส์ชื่อ Tim Berners-Lee โดยการนำเสนองานวิจัยเรื่อง prototyped ENQUIRE และ Hypertext system ซึ่งเป็นรูปแบบคำสั่งสำหรับการแบ่งปันข้อมูลระหว่างนักวิจัยภายในสถาบันวิจัยนั้น และถูกพัฒนาเรื่อยๆจนถึงปัจจุบัน

HTML ย่อมาจาก Hypertext Markup Language เป็นภาษาหลักที่ใช้ในการแสดงผลบนเว็บบราวเซอร์ภายในระบบอินเตอร์ โดยสามารถแสดงข้อมูลตัวอักษร ภาพ เสียง และไฟล์ในรูปแบบอื่นๆบนเว็บบราวเซอร์ได้อย่างมีประสิทธิภาพ โดยภาษา HTML จะแบ่งออกเป็น 2 ส่วน คือ

1. ส่วนของคำสั่ง (tag) เป็นส่วนที่กำหนดรูปแบบของข้อความที่แสดง ซึ่งเราเรียกว่า Tag โดยจะอยู่ในเครื่องหมาย < ... >

2. ส่วนของบทความ เป็นส่วนของข้อความที่แสดงผลบนเว็บบราวเซอร์

2.16.1 โครงสร้างหลักของ HTML

โครงสร้างหลักของ HTML จะเริ่มด้วย <html> และจบลงด้วย </html> เสมอ โดยมีรูปแบบโครงสร้างหลัก 2 ส่วน ได้แก่

1) head คำสั่งในส่วนนี้จะใช้กำหนดรูปแบบของหน้าเว็บไซต์ รวมถึงกำหนดรายละเอียดต่างๆเกี่ยวกับเว็บเพจ เช่น Title รูปแบบการเข้ารหัสของข้อมูล เป็นต้น

2) Body คำสั่งในส่วนนี้เป็นเนื้อหาหลักของเว็บไซต์ ประกอบไปด้วยรูปแบบตัวอักษร การจัดการรูปภาพ การเชื่อมโยงไปยังหน้าเว็บไซต์อื่น เป็นต้น

2.17 หลักการสร้างส่วนต่อประสานกราฟิกกับผู้ใช้ [17]

ทำการออกแบบส่วนต่อประสานกราฟิกกับผู้ใช้ ให้มีการแสดงผลพิกัดตำแหน่งของรถทางเกษตรกรรม เส้นทางการเดินรถ และค่าตัวแปรที่ใช้สำหรับบอกตำแหน่ง ซึ่งประกอบด้วย ทิศในการเคลื่อนที่เข้าสู่เส้นทาง และระยะห่างระหว่างสถานีจลน์กับเส้นทาง

2.17.1 การวาดเส้นบน Canvas

ในการเขียนส่วนต่อประสานกราฟิกกับผู้ใช้ จะใช้ Canvas ในการสร้างเส้นต่างๆ โดยการเขียนเส้นของ Canvas จะใช้งานฟังก์ชันบนไลบรารี่ tkinter ของภาษาไพธอน เพื่อจำลองการสร้างเส้นทางการเคลื่อนที่ของสถานีจลน์ ซึ่งการสร้างเส้นทางนั้นจำเป็นต้องใช้พิกัดจุดต้นทาง และจุดปลายทาง รวมถึงการระบุตำแหน่งต่างๆ บน canvas ดังแสดงในรูปที่ 2.27

รูปที่ 2.27 จุดพิกัดบน Canvas

จากรูปที่ 2.26 ในแผนภาพของ Canvas จุดซ้ายบนจะเป็นพิกัด (0,0) และจุดขวาล่างจะเป็นพิกัด (x,y) ตามขนาดของ Canvas ที่กำหนด เพื่อที่จะสามารถสร้างกราฟิกต่างๆ บน Canvas ได้

2.17.2 การแสดงรูปภาพบน Canvas

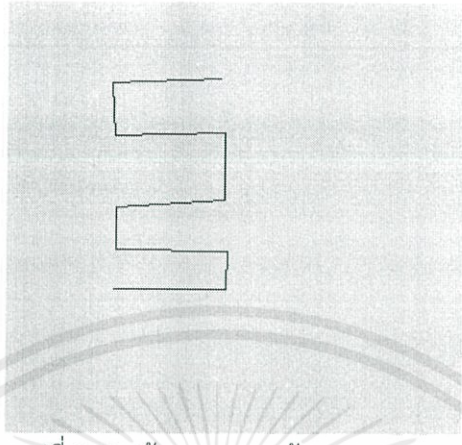
ในการวาดรูปรถทางเกษตรกรรมบน Canvas จะใช้การวาดเส้นในแต่ละพิกัดตำแหน่ง ซึ่งใช้สัญลักษณ์รูปรถแทนสถานีจลน์ ดังรูปที่ 2.28

รูปที่ 2.28 รูปรถทางเกษตรกรรมที่ใช้แทนที่ตำแหน่งสถานีจลน์

จากรูปที่ 2.28 แสดงตำแหน่งรูปรถให้อยู่ในจุดพิกัดที่กำหนดไว้เสมอ จากการสร้างเส้นเชื่อมตามแนวจุดพิกัดบน Canvas ซึ่งจะมีการใช้งานร่วมกับเส้นทาง และการบอกทิศทางใน Canvas ต่อไป

2.17.2 การวาดเส้นทางหลายเส้นบน Canvas

ในการวาดแผนที่บน Canvas จำเป็นต้องทราบค่าระยะห่างของแกน x และแกน y โดยทำการป้อนตำแหน่งในแกนสองมิติจำนวน 1 คู่ เพื่อแสดงเส้นตรง 1 เส้น ในการวาด Canvas ที่มีเส้นตรงหลายเส้นต่อกันน้ำ จะใช้ for loop ในการวาดเส้นที่ต่อเนื่องกันไปให้พิกัดบน Canvas ทั้งหมดถูกวาดลงบนส่วนต่อประสานกราฟิกกับผู้ใช้ ดังรูปที่ 2.29

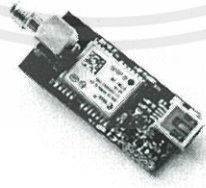


รูปที่ 2.29 เส้นทางหลายเส้นบน Canvas

จากรูปที่ 2.29 แสดงเส้นหลายเส้นบน Canvas ที่สร้างมาจากจุดพิกัดในแกน x,y ที่ได้รับจากอุปกรณ์สถานีจลณ์

2.18 เครื่องรับสัญญาณจีเอ็นเอสเอสแบบความถี่เดียวยี่ห้อ U-blox NEO M8T [18]

U-blox NEO M8T เป็นเครื่องรับสัญญาณที่รองรับการใช้งานในย่านความถี่ L1 (ประมาณ 1,575 MHz.) ซึ่งเป็นย่านกระจายสัญญาณจีเอ็นเอสเอสที่พลเรือนสามารถใช้งานได้ โดยสามารถรับสัญญาณจีเอ็นเอสเอสได้หลายระบบดาวเทียม เช่น ระบบ GPS ของอเมริกา ระบบQZSS ของญี่ปุ่น ระบบ IRNSS ของอินเดีย ระบบ GLONASS ของรัสเซีย ระบบ BeiDou ของจีนและระบบ SBAS ที่ใช้ทางการบิน และมีระบบกรองสัญญาณรบกวนแบบแถบผ่านลักษณะของเครื่องรับอยู่ในรูปที่ 2.30



รูปที่ 2.30 เครื่องรับสัญญาณจีเอ็นเอสเอสแบบความถี่เดียวยี่ห้อ U-blox NEO M8T

2.19 สายอากาศ TW-3740 [19]

สายอากาศที่ใช้ในโครงงานนี้ เป็นสายอากาศประสิทธิภาพดีรองรับการใช้งานสัญญาณจีเอ็นเอสเอสในย่านความถี่ L1 ได้หลายระบบ เช่นระบบ GPS ของสหรัฐอเมริกา ระบบ GLONASS ของรัสเซีย ระบบ QZSS ของญี่ปุ่น ระบบ Compass ของสาธารณรัฐประชาชนจีน ระบบ IRNSS ของอินเดีย และระบบ SBAS สามารถลดผลรบกวนจากสัญญาณพหุวิถีได้ดี มีอัตราขยายสัญญาณที่สูง ลักษณะของสายอากาศนี้ จะประกอบด้วยวงแหวนโลหะเป็นจำนวนหลายวงล้อมรอบสายอากาศหลัก ดังรูปที่ 2.31

รูปที่ 2.31 สายอากาศรุ่น TW-740

จากรูปที่ 2.31 เมื่อมีสัญญาณพหุวิถีเข้ามาจะถูกเหนี่ยวนำโดยวงแหวนโลหะที่ล้อมรอบสายอากาศหลัก แล้ววิ่งลงระนาบกราวนด์ของสายอากาศ ในทางการสำรวจ มักจะมีการใช้สายอากาศรูปแบบนี้มาก เนื่องจากสามารถลดผลรบกวนจากสัญญาณพหุวิถีได้ดี

2.20 ภาษาไพธอน [20]

ภาษาไพธอน (Python) ถือกำเนิดขึ้นในปี ค.ศ.1980 ถูกพัฒนาขึ้นโดย Guido vanRossum ที่ Centrum Wiskunde & Informatica (CWI) ประเทศเนเธอร์แลนด์ เป็นภาษาหนึ่งที่นิยมใช้ในการเขียนโปรแกรม ถูกพัฒนามาโดยไม่ยึดติดกลับแพลตฟอร์ม นั่นหมายความว่าภาษาไพธอนสามารถใช้งานได้ในหลายระบบปฏิบัติการ ทั้ง Unix Linux และ Windows อีกทั้งยังเป็น Open Source ทำให้นักพัฒนาโปรแกรมสามารถใช้ภาษาไพธอนมาพัฒนาโปรแกรมหรือซอฟต์แวร์ของตัวเองได้โดยไม่เสียค่าใช้จ่ายแต่อย่างใด ภาษาไพธอนเป็นภาษาระดับสูง จึงทำให้เขียนง่าย ผู้ใช้งานไม่จำเป็นต้องมีความรู้ทางด้านคอมพิวเตอร์อย่างลึกซึ้ง เพียงแค่เข้าใจไวยากรณ์และขั้นตอนการทำงานของภาษาไพธอนก็สามารถพัฒนาโปรแกรมด้วยภาษาไพธอนได้แล้ว แต่เนื่องจากความเป็น

ภาษาระดับสูงที่มีข้อดีคือเขียนและพัฒนาง่ายแล้วก็จะมีข้อจำกัดอยู่ข้อหนึ่ง คือ จะทำงานหรือประมวลผลได้ช้าเมื่อเปรียบเทียบกับภาษาอื่น เช่น ซี หรือ ฟอร์แทรน

2.20.1 ส่วนประกอบในการทำงานของภาษาไพธอน

- คอมไพเลอร์ (Compiler) ใช้ตีความคำสั่งที่ผู้พัฒนาโปรแกรมเขียนขึ้นเพื่อสั่งให้คอมพิวเตอร์ทำงาน

- อีดิเตอร์ (Editor) ใช้เขียนชุดคำสั่งเพื่อให้คอมไพเลอร์อ่านแล้วสั่งคอมพิวเตอร์อีกครั้งหนึ่ง

- เชลโต้ตอบ (interactive shell) หรือ อินเทอร์พรีเตอร์ (interpreter) ใช้ป้อนคำสั่งเพื่อสั่งการคอมพิวเตอร์แบบทันที

2.20.2 ข้อดีของภาษาไพธอน

- ไพธอนเป็นภาษาสคริปต์ ทำให้ใช้เวลาในการเขียนและคอมไพล์ไม่มากเหมาะกับงานด้านการดูแลระบบ (System administration)

- คำสั่งที่เขียนด้วยภาษาไพธอนสามารถนำไปทำงานบนระบบปฏิบัติการได้หลากหลาย

- การประมวลผลจะทำในแบบอินเทอร์พรีเตอร์ คือ จะประมวลผลไปทีละบรรทัดและปฏิบัติตามคำสั่งที่ได้รับ

- เป็นภาษาแบบ Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวกรวดเร็ว

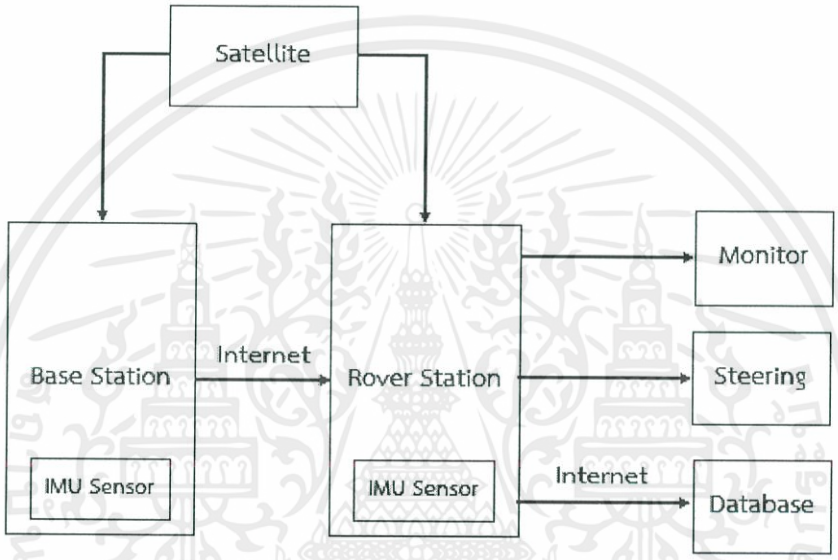
- มี Build-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ใน Python ซึ่งประกอบไปด้วย ลิสต์ ดิกชันนารี สตริง ที่ง่ายต่อการใช้งานและมีประสิทธิภาพสูง มีฟังก์ชันสนับสนุนฐานข้อมูล เช่น MySQL Sybase Oracle Informix ODBC และอื่นๆ

- มีไลบรารีสนับสนุนด้านปัญญาประดิษฐ์

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ



รูปที่ 3.1 บล็อกไดอะแกรมแสดงภาพรวมของโครงงาน

จากรูปที่ 3.1 แสดงภาพรวมของโครงงาน ซึ่งเป็นระบบการเฝ้าสังเกตและติดตามตำแหน่งของสถานีจลน์ (Rover) ให้มีความแม่นยำ เพื่อสร้างระบบนำร่องรถทางเกษตรกรรมที่จะแจ้งเตือนผู้ใช้งานผ่านทางหน้าจอแสดงผล ให้สามารถเดินรถทางเกษตรกรรมไปตามเส้นทางที่กำหนดไว้ได้ โดยใช้เทคนิค RTK (Real Time Kinematic) ซึ่งประกอบไปด้วยสถานีฐาน (Base station) และสถานีจลน์ ซึ่งทั้งสองอุปกรณ์ต้องตั้งอยู่ในพื้นที่ที่สามารถมองเห็นดาวเทียมดวงเดียวกันได้ ตลอดจนมีการเชื่อมต่อกันผ่านโครงข่ายอิเทอร์เน็ต (Internet) จากนั้นสถานีฐานจะมีการส่งค่าแก้ไปยังสถานีจลน์ เพื่อช่วยลดความผิดพลาดของสัญญาณจีเอสเอสในชั้นบรรยากาศ นิยมใช้งานรังวัด (survey) รวมถึงการเกษตรแม่นยำสูง

ในโครงงานนี้จะใช้ Raspberry Pi ในการประมวลผลค่าสัญญาณจีเอ็นเอสเอสจากดาวเทียมเป็นค่าพิกัดตำแหน่ง และใช้ IMU (Inertial Measurement Unit) สำหรับวัดความเอียงของสายอากาศที่ใช้รับสัญญาณจีเอ็นเอสเอส เพื่อลดความผิดพลาดในการรับสัญญาณของสายอากาศขณะออกสำรวจในภูมิประเทศจริง ตลอดจนติดตั้งหน้าจอสัมผัสสำหรับตั้งค่าการใช้งาน รวมไปถึงการติดตามตำแหน่งของรถทางเกษตรกรรม และการควบคุมการเคลื่อนที่ของรถทางเกษตรกรรมผ่าน

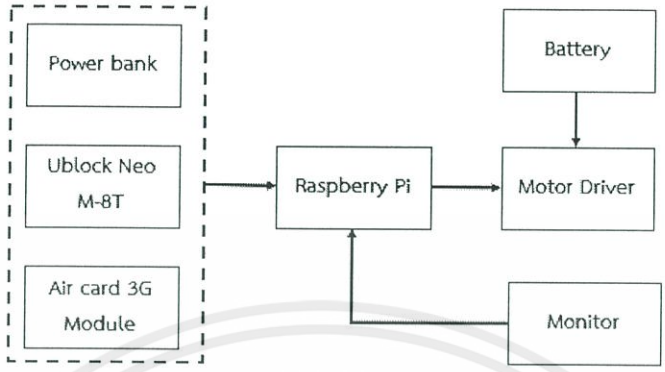
ระบบควบคุมการเคลื่อนที่อัตโนมัติ (Steering) นอกจากนี้ยังมีการส่งผ่านข้อมูลพิกัดตำแหน่ง และการทำงานของระบบไปยังฐานข้อมูล (Database) แล้วแสดงผลข้อมูลจากฐานข้อมูลผ่านหน้าเว็บไซต์ เพื่อให้ผู้ใช้งานสามารถตรวจสอบการทำงานย้อนหลังของสถานีจลน์ได้

3.1.1 การออกแบบสถานีจลน์

ทำการออกแบบอุปกรณ์สถานีจลน์เพื่อใช้ในการทดสอบ RTK ในการสร้างสถานีจลน์นั้น จะต้องมีการกำหนดให้มีขนาด และความทนทานให้เหมาะสม ตลอดจนไปใช้งานได้อย่างสะดวก โดยมีองค์ประกอบแผนภาพการดำเนินงาน ดังรูปที่ 3.2

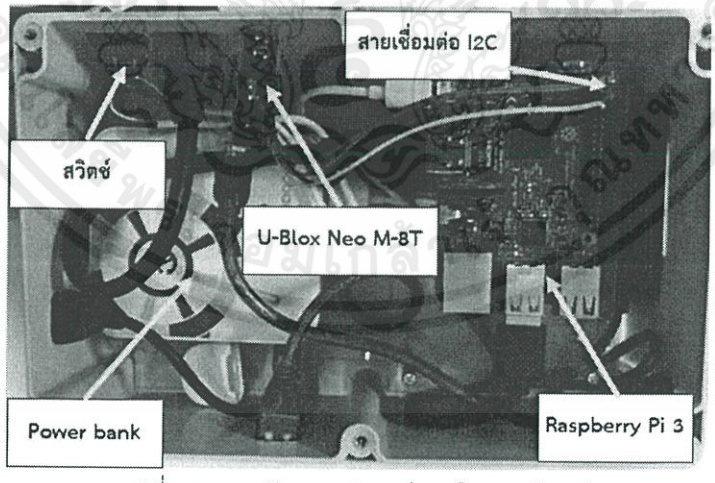
รูปที่ 3.2 การประกอบสถานีจลน์

จากรูปที่ 3.2 แสดงการออกแบบ และประกอบอุปกรณ์สถานีจลน์ โดยเริ่มต้นจากการวัดขนาดของอุปกรณ์ภายในสถานีจลน์ ต่อมาเลือกอุปกรณ์ที่มีขนาดเหมาะสมในการใช้งาน และสุดท้ายทำการติดตั้งอุปกรณ์ให้มีความเชื่อมโยงกันอย่างเป็นระเบียบภายในกล่องสถานีจลน์

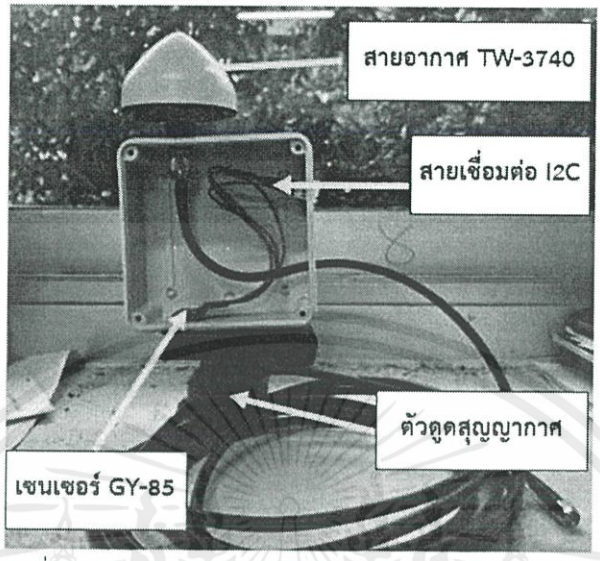


รูปที่ 3.3 แผนภาพอุปกรณ์สถานีจลน์

จากรูปที่ 3.3 อุปกรณ์สถานีจลน์ ประกอบไปด้วย Raspberry Pi และ U-Blox NEO-M8T ซึ่งเป็นส่วนที่ใช้ระบุพิกัดสถานีจลน์ โดย U-Blox Neo M-8T จะเชื่อมต่อกับสายอากาศ TW-3740 ที่ใช้รับสัญญาณจีเอ็นเอสเอส และเซ็นเซอร์ GY-85 สำหรับอ่านค่ามุมเอียงของสายอากาศ ได้แก่ มุม Roll Pitch และ Yaw มีการบรรจุไอซี L298N และแบตเตอรี่ เพื่อใช้ในการควบคุมการหมุนของมอเตอร์สำหรับการควบคุมรถทางเกษตรกรรมจำลอง จากนั้นแสดงผลผ่านหน้าจอแสดงผล LCD แบบสัมผัสขนาด 7 นิ้ว โดยใช้แหล่งจ่ายไฟเป็น Power Bank ขนาดความจุ 20000 mAh รวมถึง ดังแสดงในรูปที่ 3.4 รูปที่ 3.5 และรูปที่ 3.6

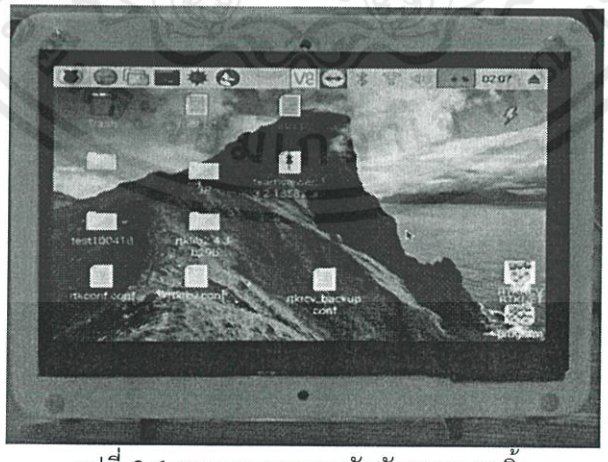


รูปที่ 3.4 การจัดวางอุปกรณ์ภายในสถานีจลน์



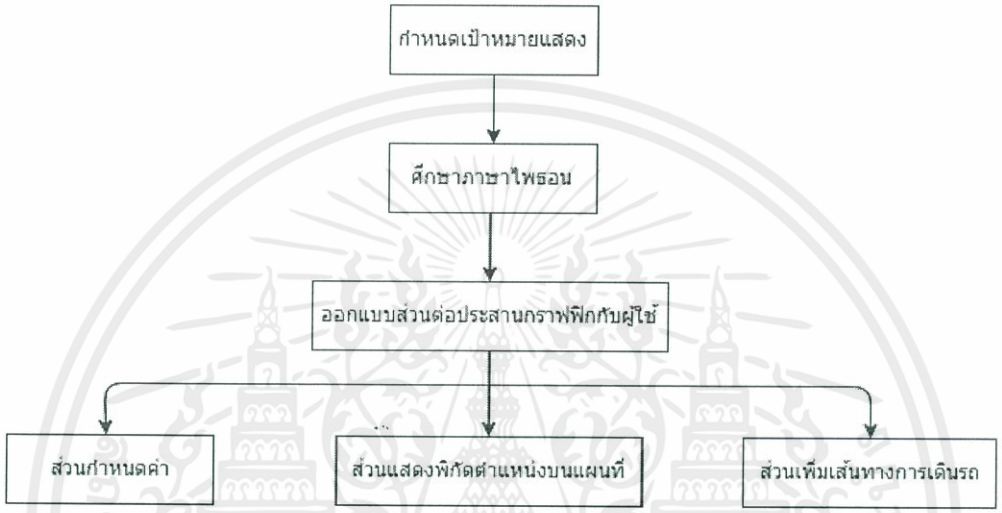
รูปที่ 3.5 สถานีจลน์ส่วนสายอากาศ และเซนเซอร์ GY-85

จากรูปที่ 3.4 และรูปที่ 3.5 แสดงอุปกรณ์ภายในสถานีจลน์ ซึ่งมีการเชื่อมต่อกันเพื่อประมวลผลหาค่าพิกัดสำหรับรถทางเกษตรกรรม โดยเริ่มจากการรับสัญญาณจีเอ็นเอสเอสจากสายอากาศ TW-3740 แล้วส่งส่งค่าพิกัดไปยัง U-Blox เพื่อแปลงสัญญาณที่ได้ให้อยู่ในรูปของข้อมูลแบบดิจิตอล แล้วนำไปประมวลผลบน Raspberry Pi เมื่อประมวลผลเสร็จสิ้นจะแสดงค่าต่างๆ ที่ใช้ในการเดินรถทางเกษตรกรรมบนหน้าจอแสดงผลขนาด 7 นิ้ว ผ่านส่วนต่อประสานกราฟิกกับผู้ใช้ ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 จอแสดงผลแบบสัมผัสขนาด 7 นิ้ว

จากรูปที่ 3.6 บนหน้าจอสำหรับผู้ใช้งานบนสถานีจลน์ จะทำการออกแบบหน้าต่างการใช้งานหลักของส่วนต่อประสานกราฟิกกับผู้ใช้ ให้มีการติดตามการแสดงพิกัดทางหน้าจอ และการนำทางในเส้นทางที่กำหนดไว้พร้อมทั้งแจ้งเตือนเมื่อออกเส้นทาง ดังรูปที่ 3.7



รูปที่ 3.7 ขั้นตอนการออกแบบหน้าต่างหลักของส่วนต่อประสานกราฟิกกับผู้ใช้

จากรูปที่ 3.7 แสดงขั้นตอนการออกแบบหน้าต่างหลักของส่วนต่อประสานกราฟิกกับผู้ใช้ โดยต้องการแสดงผลพิกัดตำแหน่งบนแผนที่ มีส่วนแจ้งเตือนเมื่อออกนอกเส้นทางที่กำหนด ส่วนการตั้งค่าการรับค่าพิกัดให้ได้ตามลักษณะการใช้งาน และส่วนออกแบบเพื่อทำการเพิ่มและปรับแต่งเส้นทางตามลักษณะการใช้งาน โดยใช้ภาษา Python ซึ่งเป็นภาษาที่ง่ายต่อการทำความเข้าใจ และสามารถแสดงส่วนต่อประสานกราฟิกกับผู้ใช้ได้

3.1.1.1 การออกแบบหน้าต่างการ Configuration ของระบบ

ปกติแล้วการ Configuration ของระบบในแต่ละครั้ง ผู้ใช้งานจะต้องพิมพ์คำสั่งลงบน Command Line เพื่อแก้ไขไฟล์ Configuration แล้วให้โปรแกรม RTKRCV ประมวลผลในการรับค่าพิกัดตำแหน่ง ดังรูปที่ 3.14 ซึ่งก่อให้เกิดความไม่สะดวกในการใช้งาน จึงจำเป็นต้องออกแบบและสร้างหน้าต่าง Configuration ขึ้นมา เพื่อให้ผู้ใช้งานสามารถใช้งานระบบได้อย่างรวดเร็ว และง่ายต่อความเข้าใจ โดยไฟล์ Config ของโปรแกรม RTKRCV จะแสดงดังรูป 3.8

```
# rtkrcv options (4/09/2018 , v.2.4.3b29d)
# if want tcpsvr:22102/
console-passwd     =admin
console-timetype   =utc        # (0:gpst,1:utc,2:jst,3:tow)
console-soltype    =dms        # (0:dms,1:deg,2:xyz,3:enu,4:pyl)
console-solflag    =1          # (0:off,1:std+2:age/ratio/ns)
inpstr1-type       =serial     # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr2-type       =ntripcli     # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr3-type       =#ftp       # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr1-path       =ttyACM0:115200:8:n:1:off
inpstr2-path       =tele4:csssrg@cssrg-pc.telecom.kmitl.ac.th:2101/KMIT6:
inpstr3-path       =ftp://anonymous:passwd@cddis.gsfc.nasa.gov/gps/products/%w/igu%w%D_%hb.sp3.Z::T=-14400,21600,7200,600
inpstr1-format     =ubx        # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,14:sp3)
inpstr2-format     =rtcm3        # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,14:sp3)
inpstr3-format     =#sp3         # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,14:sp3)
inpstr2-nmeareq    =off        # (0:off,1:latlon,2:single)
inpstr2-nmealat    =0          # (deg)
inpstr2-nmealon    =0          # (deg)
outstr1-type       =tcpsvr       # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
outstr2-type       =tcpsvr       # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
outstr1-path       =:@:2017/:
outstr2-path       =:@:2016/:
outstr1-format     =llh        # (0:llh,1:xyz,2:enu,3:nmea)
outstr2-format     =llh        # (0:llh,1:xyz,2:enu,3:nmea)
logstr1-type       =off#file       # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr2-type       =off#file       # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr3-type       =off        # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr1-path       =rov_%Y%m%d%h%M.log
logstr2-path       =base_%Y%m%d%h%M.log
logstr3-path       =cor_%Y%m%d%h%M.log
misc-svrcycle      =10         # (ms)
misc-timeout       =30000      # (ms)
misc-reconnect     =30000      # (ms)
misc-nmeacycle     =5000       # (ms)
misc-buffsize      =32768      # (bytes)
misc-navmsgsel     =rover      # (0:all,1:rover,1:base,2:corr)
misc-startcmd      =/home/pi/Desktop/program/RTKLIB-demo5/app/rtkrcv/gcc/rtkstart.sh
misc-stopcmd       =/home/pi/Desktop/program/RTKLIB-demo5/app/rtkrcv/gcc/rtkshut.sh
```

รูปที่ 3.8 ไฟล์ Config ของโปรแกรม RTKRCV

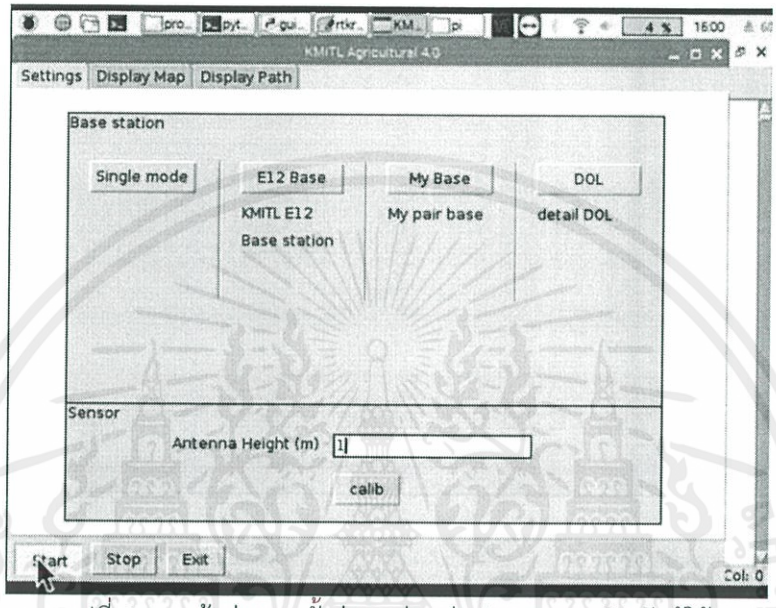
จากรูปที่ 3.8 แสดงการตั้งค่าพารามิเตอร์ต่างๆ ในการใช้งานโปรแกรม RTKRCV โดยผู้จัดทำเลือกค่าพารามิเตอร์ที่ต้องตั้งค่าเป็นประจำ ดังตัวอย่างในกรอบสีแดงจากรูปที่ 3.8 ไปทำการสร้างหน้าต่างส่วนต่อประสานกราฟิกผู้ใช้เพื่อแก้ไขค่าพารามิเตอร์ดังกล่าว โดยปรับรูปแบบการตั้งค่าให้เหมาะสมและเป็นมิตรกับผู้ใช้ คือสามารถเลือกโหมดของการระบุตำแหน่ง (Positioning Mode) ได้ 3 แบบ คือ

1. single เป็นโหมดการคำนวณหาพิกัดของสถานีจลน์โดยไม่พึ่งค่าแก้ (หรือ Correction) เป็นค่าความผิดพลาดในชั้นบรรยากาศที่มีผลกระทบต่อการเดินทางของสัญญาณจีเอ็นเอสเอส) ของสถานีฐานใดๆ เพื่อเพิ่มความแม่นยำให้กับตนเอง การใช้โหมดนี้ ในกรณีที่ใช้กับเครื่องรับสัญญาณจีเอ็นเอสเอสชนิดความถี่เดียว และ 2 ความถี่ จะมีความแม่นยำประมาณ 15 เมตร และ 5 เซนติเมตรตามลำดับ

2. static เป็นโหมดการคำนวณหาพิกัดของสถานีจลน์ขณะอยู่นิ่งกับที่ โดยรับค่าแก้จากสถานีฐาน เพื่อเพิ่มความแม่นยำในการระบุตำแหน่งถึงระดับไม่เกิน 2 เซนติเมตร โดยทั่วไปจะใช้โหมดนี้ในการปักหมุดเส้นทางเดินรถหรือสร้างสถานีฐานในบริเวณพื้นที่ทดสอบเพื่อลดความคลาดเคลื่อนในการระบุพิกัด

3. kinematic เป็นโหมดการคำนวณหาพิกัดของสถานีจลน์ขณะเคลื่อนที่ โดยรับค่าแก้จากสถานีฐานเพื่อเพิ่มความแม่นยำให้กับตนเอง ซึ่งจะให้ความแม่นยำของการระบุตำแหน่งในระดับ 1 เซนติเมตร ถึง 5 เซนติเมตร มักใช้ในการติดตามการเคลื่อนที่ของรถ เพื่อประกอบการควบคุม และนำทางรถทางเกษตรกรรม

ในการออกแบบได้สร้างปุ่มที่สามารถเลือกได้สถานีฐานได้ เพื่อให้เหมาะสมกับสถานที่ที่ใช้งาน ดังรูปที่ 3.9



รูปที่ 3.9 หน้าต่างการตั้งค่าบนส่วนต่อประสานกราฟิกกับผู้ใช้

จากรูปที่ 3.9 แสดงหน้าต่างของส่วนการตั้งค่า โดยประกอบไปด้วยสองส่วนคือ การเลือกสถานีฐาน และส่วนการปรับเทียบเซ็นเซอร์ โดยมีปุ่มในการปรับแต่งส่วนการตั้งค่า 5 ส่วนมีดังนี้

1. Single mode เมื่อกดใช้ปุ่มนี้ จะเป็นการรับสัญญาณจีเอ็นเอสเอส ด้วยเครื่องรับสัญญาณจีเอ็นเอสเอสชนิดความถี่เดียวโดยไม่จำเป็นต้องพึ่งพาการแก้ไขค่าจากเทคนิค RTK

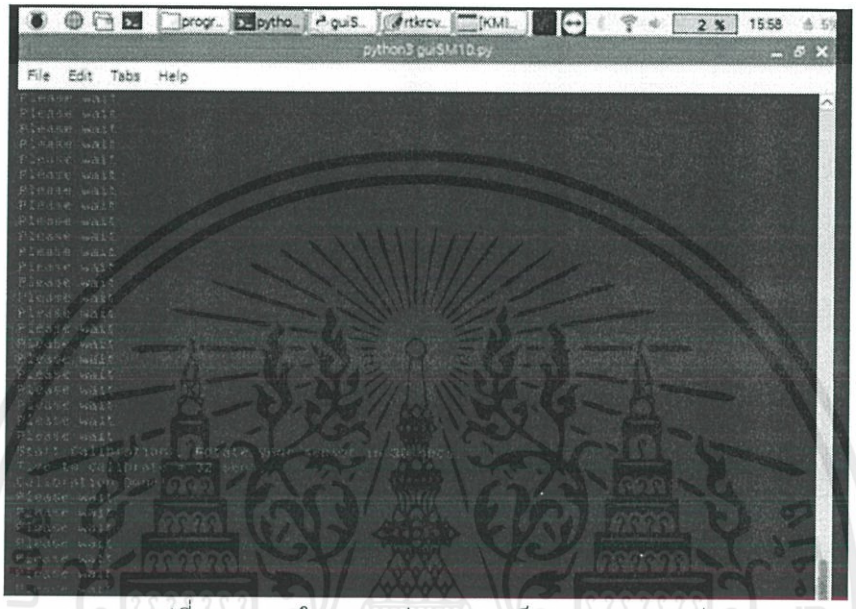
2. E12 Base เมื่อกดใช้ปุ่มนี้ จะเป็นการรับค่าจากสถานีฐานชนิดสองความถี่เพื่อนำมาใช้ในเทคนิค RTK ที่อาคาร 12 ชั้นคณะวิศวกรรมศาสตร์ สถานบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ใช้ในกรณีที่การทดสอบอยู่ในรัศมีระยะทางไม่ถึง 1 กิโลเมตรจากอาคาร 12 ชั้น

3. My Base เมื่อกดใช้ปุ่มนี้ จะเป็นการรับค่าจากสถานีฐานที่สร้างขึ้นมาเองจากสถานีชนิดหนึ่งความถี่เพื่อนำมาใช้ในเทคนิค RTK ใช้ในกรณีที่ทดสอบอยู่ในรัศมีระยะทางไม่เกิน 1 กิโลเมตร

4.DOL หรือ Department of Lands เมื่อกดใช้ปุ่มนี้ จะเป็นการรับค่าจากสถานีฐานชนิดสองความถี่เพื่อนำมาใช้ในเทคนิค RTK

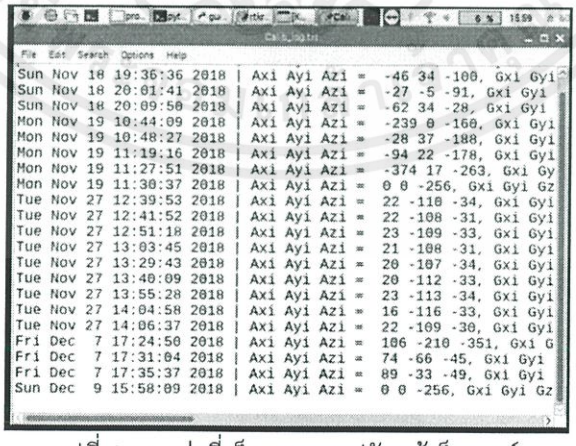
เมื่อโปรแกรมได้รับค่าอินพุตจากการป้อนโดยผู้ใช้งานแล้ว ชุดคำสั่งภายในโปรแกรมจะนำค่าอินพุตเหล่านั้นไปแก้ไขไฟล์ rtkrcv.conf แล้วให้โปรแกรม RTKRCV นำไปใช้ประมวลผลในการรับค่าพิกัดตำแหน่งต่อไปในส่วนของการออกแบบหน้าต่างการปรับเทียบ (Calibrate) เซ็นเซอร์ GY-85 ในการออกแบบได้ทำการแบ่งแยกหน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้เป็นส่วนๆ โดยหากทำ

การต่อเซ็นเซอร์ จะต้องป้อนความสูงของสายอากาศในหน่วยเมตร วัดจากพื้นดิน และทำการปรับแก้ โดยการกดปุ่ม calib ซึ่งจะได้ผลลัพธ์ดังแสดงในรูปที่ 3.10



รูปที่ 3.10 ผลในการกดปุ่ม calib เป็นเวลา 30 วินาที

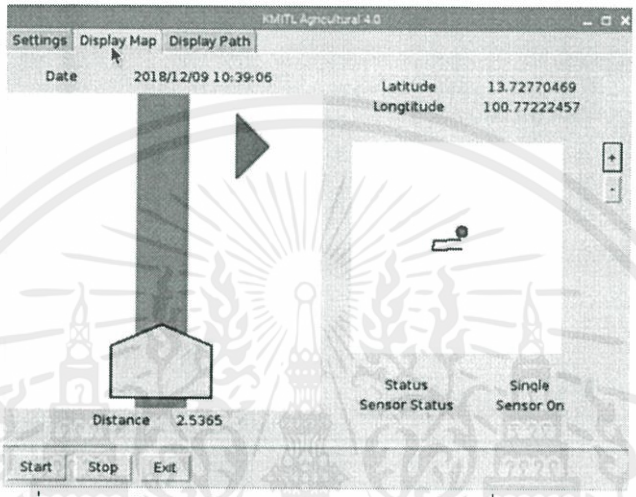
จากรูปที่ 3.10 แสดงผลลัพธ์ในการปรับแก้เซ็นเซอร์ โดยจะต้องทำการหมุนหรือขยับเซ็นเซอร์เป็นเวลาประมาณ 30 วินาที เพื่อเป็นการกำหนดการเริ่มต้นทำงานของเซ็นเซอร์ โดยจะต้องทำทุกครั้งที่เปลี่ยนสถานที่ในการติดตั้ง จะได้ผลลัพธ์ในการปรับแก้ ในไฟล์ .txt ดังรูปที่ 3.11



รูปที่ 3.11 ค่าที่เก็บจากการปรับแก้เซ็นเซอร์

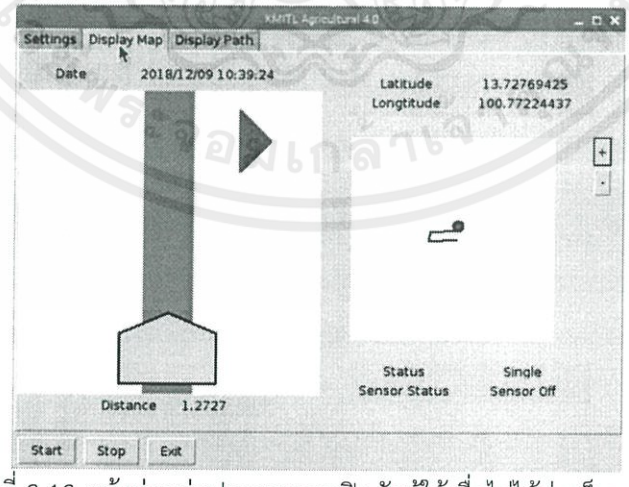
จากรูปที่ 3.11 แสดงถึงค่าที่เกิดจากการหมุนเซ็นเซอร์เป็นเวลา 30 วินาที เพื่อนำไปเป็นตัวแปรในการปรับแก้ค่าความเอียงของสายอากาศ ให้พิกัดตำแหน่งมีความถูกต้องมากขึ้น

ในกรณีที่ไม่ได้มีการต่อเซ็นเซอร์ ระบบก็ยังสามารถทำงานได้ โดยจะไม่มีการใช้การแก้พิกัดจากเซ็นเซอร์เท่านั้น หากเซ็นเซอร์ที่ต่ออยู่หลุดระหว่างการทำงาน โปรแกรมยังสามารถแสดงผลได้อย่างต่อเนื่อง ดังรูปที่ 3.12



รูปที่ 3.12 หน้าส่วนต่อประสานกราฟิกกับผู้ใช้เมื่อต่อเซ็นเซอร์

จากรูป 3.12 เมื่อได้ต่อเซ็นเซอร์สถานะของเซ็นเซอร์จะขึ้นว่า Sensor On โดยระบบจะมีการปรับแก้พิกัดตำแหน่งโดยอาศัยค่ามุมจากเซ็นเซอร์ และค่าความสูงของสายอากาศที่กรอกไว้ในส่วนของการตั้งค่า และหากไม่ได้ต่อเซนเซอร์จะแสดงผลการทำงานของโปรแกรม ดังรูปที่ 3.13

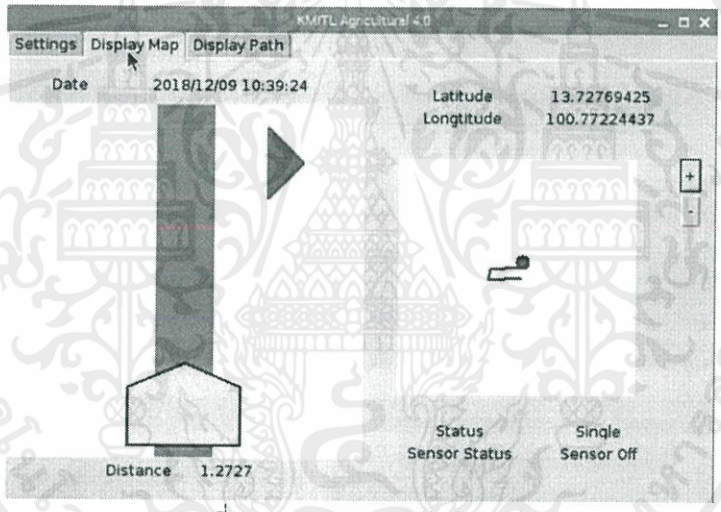


รูปที่ 3.13 หน้าส่วนต่อประสานกราฟิกกับผู้ใช้เมื่อไม่ได้ต่อเซ็นเซอร์

จากรูป 3.13 เมื่อไม่ได้ต่อเซ็นเซอร์หรือเซนเซอร์หลุดออก จะแสดงสถานะของเซ็นเซอร์ด้วยคำว่า Sensor จากนั้นระบบจะปรับการทำงานให้สามารถใช้งานโปรมแกรมต่อได้ โดยไม่มีผลกระทบกับการทำงานของโปรแกรม

3.1.1.2 การแสดงพิกัดตำแหน่งบนแผนที่ Canvas

ในส่วนนี้จะทำการรับค่าพิกัดจากพอร์ตของ Raspberry Pi ที่เชื่อมต่อกับตัวรับสัญญาณ U-Blox โดยนำมากรองส่วน ละติจูด และลองจิจูด มาแสดงผลในแผนที่ Canvas แบบเวลาจริง และแบบจำลองแผนที่ที่สร้างขึ้นเองซึ่งจะแสดงค่าละติจูด และลองจิจูดแบบเวลาจริง หากมีการเคลื่อนที่ แผนที่จะเปลี่ยนไปตามการเคลื่อนที่ และแสดงตำแหน่งปัจจุบันเพื่อนำทางบนส่วนกราฟิกประสานงานกับผู้ใช้ ดังรูปที่ 3.14



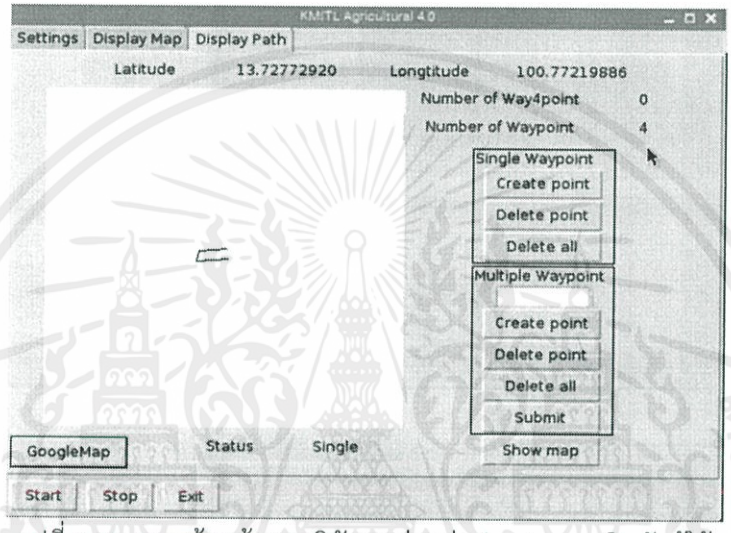
รูปที่ 3.14 การแสดงพิกัดตำแหน่ง

จากรูปที่ 3.14 แสดงพิกัดตำแหน่งปัจจุบันและจะเห็นได้ว่าพิกัดปัจจุบันอยู่ที่ละติจูด 13.72769425 และลองจิจูด 100.77224437 ซึ่งอยู่ห่างจากเส้นทาง 1.2727 เมตร และไม่มีการใช้เซนเซอร์ปรับแก้พิกัดจากความเอียงตัวของสายอากาศ

3.1.1.3 การสร้างพิกัดเส้นทางบนส่วนต่อประสานกราฟิกกับผู้ใช้

ประกอบด้วยการสร้างพิกัดสองแบบ คือการเก็บเส้นทางพิกัดแบบSingle Waypoint และแบบ Multiple Waypoint แบบ Single Waypoint เป็นการเก็บพิกัดเส้นทางโดยการเก็บพิกัดของสถานีจลน์ในปัจจุบันโดยการเก็บที่ละจุดและนำจุดพิกัดทั้งหมดที่เก็บมาเรียงเป็นเส้นทางพิกัดโดยเรียงตามลำดับ และการเก็บเส้นทางพิกัดแบบ Multiple Waypoint เป็นการเก็บพิกัดที่ต้องใช้พิกัดหลัก 4 จุดเพื่อใช้ในการสร้างเส้นทางพิกัดขึ้นมาโดยผู้ใช้จำเป็นจะต้องกำหนดค่าจำนวนเส้นพิกัดซึ่งจำนวนค่าพิกัดจะเป็นจำนวนเส้นพิกัดที่อยู่ระหว่างจุดพิกัด 4 จุดที่ทำการเพิ่มเข้าไป

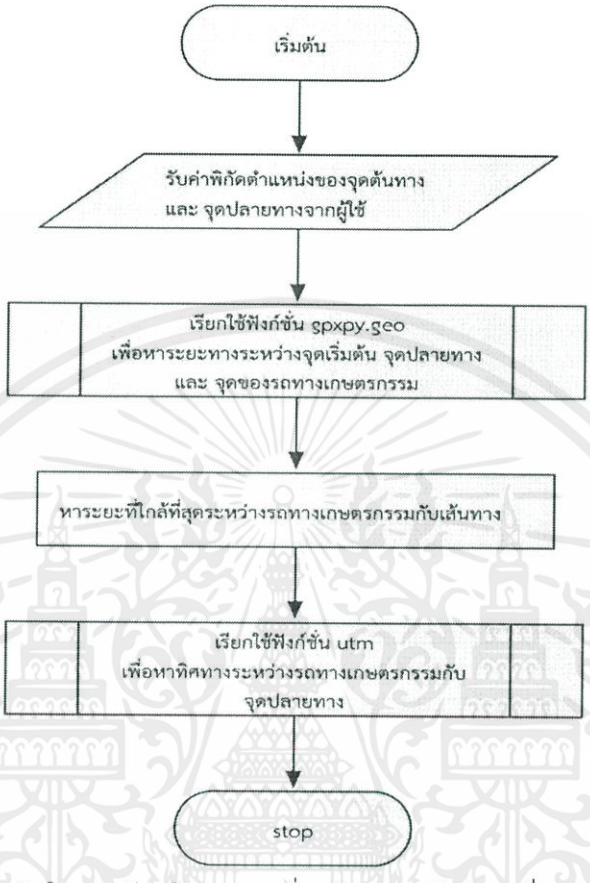
ในตอนแรก ในการรับค่าพิกัดของตำแหน่งปัจจุบันเพื่อนำไปใช้ในการหาเส้นทางพิกัดทำได้โดยการกดปุ่ม Create point โดยปุ่มนี้จะทำการรับพิกัดของตำแหน่งปัจจุบันทั้ง ละติจูด และลองจิจูด ซึ่งจะนำพิกัดในปัจจุบันไปเก็บไว้ในไฟล์ .txt และทำการสร้างแผนที่จำลองเพื่อให้ผู้ใช้สามารถปรับแต่งและสร้างเส้นทางพิกัดได้สะดวกขึ้น ดังรูปที่ 3.15



รูปที่ 3.15 การสร้างเส้นทางพิกัดบนส่วนต่อประสานกราฟิกกับผู้ใช้

จากรูปที่ 3.15 จะเห็นได้ว่าสถานะของจุดพิกัดทางด้านขวาขึ้นว่ามีจุดพิกัดทั้งหมด 4 จุด จากนั้นจึงแสดงจุดพิกัด 4 จุดออกมาเป็นเส้นทางพิกัดดังแสดงทางซ้ายมือ

3.1.1.4 แนวคิดในการเขียนโปรแกรมเพื่อสร้างส่วนแจ้งเตือนเมื่อออกนอกเส้นทาง แสดงดังรูปที่ 3.16

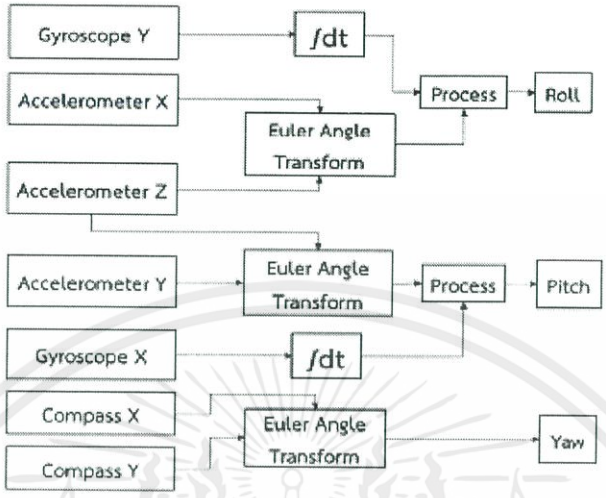


รูปที่ 3.16 แนวคิดในการเขียนโปรแกรมเพื่อสร้างส่วนแจ้งเตือนเมื่อออกนอกเส้นทาง

จากรูปที่ 3.16 เป็นทำการป้อนตำแหน่งจุดต้นทาง และปลายทางผ่านจอแสดงผล เพื่อทำการสร้างเส้นทางโดยจะเส้นทางพิกัดบนแผนที่จำลอง และนำค่าพิกัดของรถทางเกษตรกรรม มาประมวลผลโดยใช้ฟังก์ชั่น gpxpy ในการหาระยะระหว่างพิกัด เพื่อใช้ในการคำนวณหาระยะห่างระหว่างตัวรถกับเส้นทาง และใช้ฟังก์ชั่น utm ในการระบุทิศทาง ในรูปแบบของทิศทางซ้ายหรือขวา หากรถออกนอกเส้นทาง จะทำการแจ้งระยะห่างของรถทางเกษตรกรรมกับเส้นทาง ทิศทางที่ผิดพลาด

3.1.1.5 การอ่านค่ามุมที่แสดงการเคลื่อนที่ของวัตถุโดยใช้เซนเซอร์รุ่น GY-85

จากหลักการทำงานของ เซนเซอร์รุ่น GY-85 ในหัวข้อที่ 2.10 สามารถแสดงค่ามุมที่เอียงไปของวัตถุโดยแสดงดังรูป 3.17



รูปที่ 3.17 การอ่านค่ามุมแสดงการเคลื่อนที่ของวัตถุโดยใช้ เซนเซอร์ รุ่น GY-85

จากรูปที่ 3.17 ค่าต่างๆในแผนภาพ สามารถอธิบายได้ดังนี้

Accelerometer X Accelerometer Y และ Accelerometer Z คือสัญญาณขาออกตามแนวแกน X แกน Y และแกน Z ของ Accelerometer

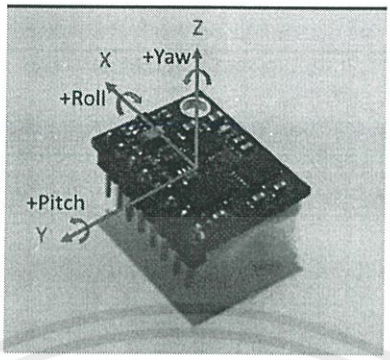
Gyroscope X Gyroscope Y และ Gyroscope Z คือสัญญาณขาออกตามแนวแกน X แกน Y และแกน Z ของ Gyroscope

Compass X และ Compass Y คือสัญญาณขาออกตามแนวแกน X แกน Y ของ Magnetometer

ค่า Roll คือมุมที่วัดรอบแกน X ของ เซนเซอร์ รุ่น GY-85 ในทิศทวนเข็มนาฬิกา มีหน่วยเป็นองศา

ค่า Pitch คือมุมที่วัดรอบแกน Y ของ เซนเซอร์ รุ่น GY-85 ในทิศทวนเข็มนาฬิกา มีหน่วยเป็นองศา

ค่า Yaw คือมุมที่วัดรอบแกน Z ของ เซนเซอร์ รุ่น GY-85 ในทิศทวนเข็มนาฬิกา มีหน่วยเป็นองศา

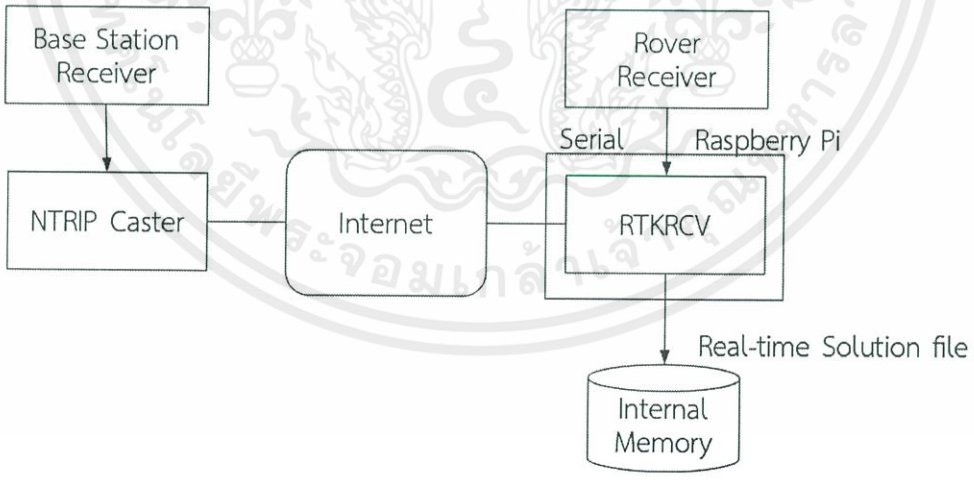


รูปที่ 3.18 แนวการวัดมุม Roll Pitch และ Yaw ของเซนเซอร์ รุ่น GY-85

จากรูปที่ 3.18 เป็นข้อตกลงในการทิศการวัดมุม Roll Pitch และ Yaw เพื่อเป็นแนวทางในการเลือกสัญญาณขาออกของ เซนเซอร์ รุ่น GY-85 มาคำนวณหามุมดังกล่าวอย่างถูกต้อง

3.1.2 การคำนวณตำแหน่งด้วยเทคนิค RTK โดยใช้โปรแกรม RTKLIB

ในโครงงานนี้ จะใช้โปรแกรม RTKRCV ซึ่งเป็นโปรแกรมย่อยในโปรแกรม RTKLIB โดยการใช้แต่ละโปรแกรมเพื่อการคำนวณตำแหน่งด้วยเทคนิค RTK สามารถแสดงรูปแบบการทำงานได้ดังรูปที่ 3.19

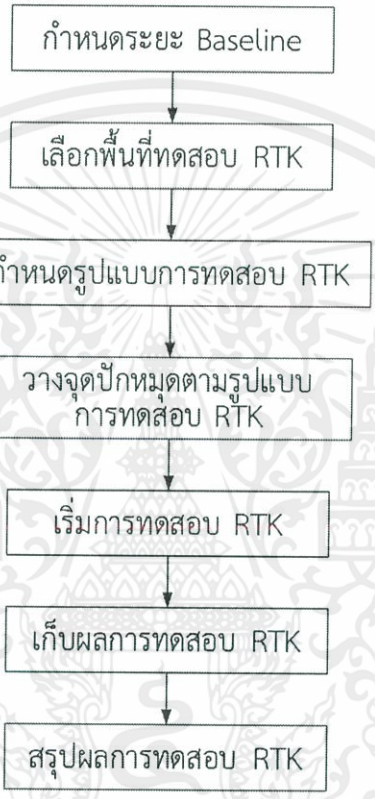


รูปที่ 3.19 การใช้โปรแกรม RTKRCV ในการคำนวณตำแหน่งด้วยเทคนิค RTK

จากรูปที่ 3.19 การคำนวณตำแหน่งด้วยเทคนิค RTK จะต้องตั้งค่าโปรแกรม RTKRCV ใน Raspberry Pi ให้รับข้อมูลดิบที่ได้จากเครื่องรับสัญญาณจีเอ็นเอสเอสสถานีฐานผ่านโครงข่าย

Internet มาประมวลผล เพื่อดูผลการคำนวณตำแหน่งด้วยเทคนิค RTK บน Raspberry Pi แบบเวลาจริง

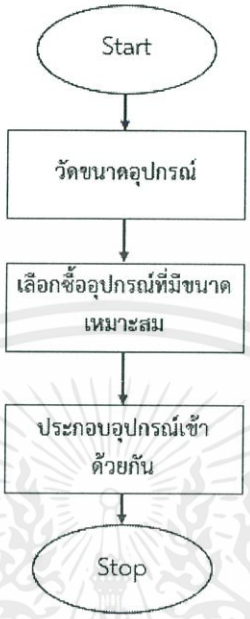
ขั้นตอนในการคำนวณตำแหน่งด้วยเทคนิค RTK สามารถแสดงได้ดังรูปที่ 3.20



รูปที่ 3.20 ขั้นตอนการทดสอบระยะ Baseline ที่มีผลต่อความแม่นยำในการทำ RTK

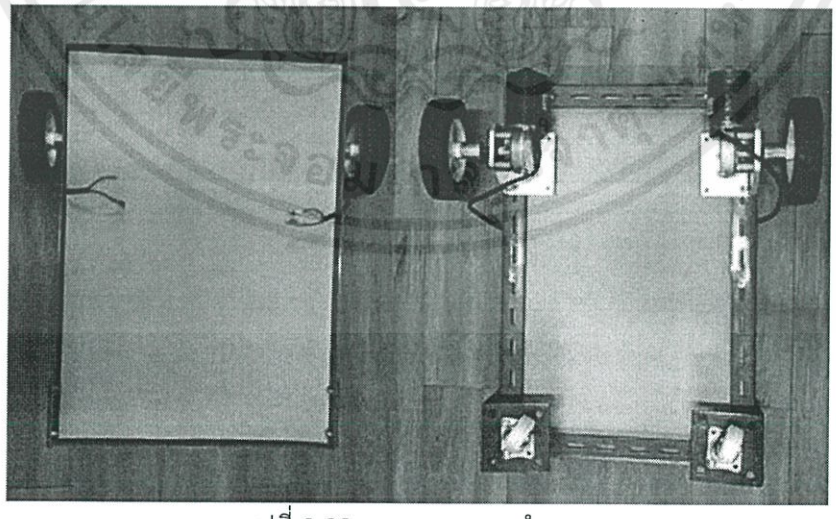
3.1.3 ออกแบบรถทางเกษตรกรรมจำลอง

ทำการออกแบบรถทางเกษตรกรรมจำลองเพื่อติดตั้งสถานีจลน์ และใช้ในการทดสอบ RTK นั้นจะต้องมีการกำหนดให้มีขนาด และความทนทานให้เหมาะสม ตลอดจนไปใช้งานได้อย่างสะดวก โดยมีองค์ประกอบแผนภาพการดำเนินงาน ดังรูปที่ 3.21



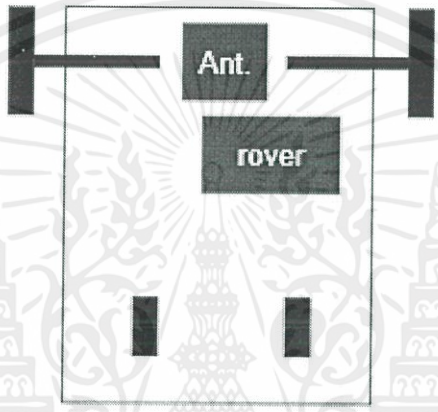
รูปที่ 3.21 การประกอบรถเกษตรกรรมจำลอง

จากรูปที่ 3.21 แสดงการออกแบบ และประกอบรถทางเกษตรกรรมจำลอง โดยเริ่มต้นจากการวัดขนาดของอุปกรณ์ ต่อมาเลือกอุปกรณ์ที่มีขนาดเหมาะสมในการใช้งาน และทำการติดตั้งอุปกรณ์ให้มีความเชื่อมโยงกันอย่างเป็นระเบียบ เพื่อให้ได้รถเกษตรกรรมจำลองตามที่ต้องการ ดังแสดงในรูปที่ 3.22



รูปที่ 3.22 รถเกษตรกรรมจำลอง

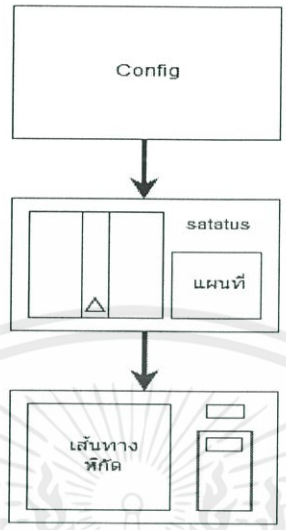
จากรูปที่ 3.22 แสดงรถทางเกษตรกรรมจำลอง โดยรูปทางด้านซ้ายเป็นด้านบนของรถเกษตรกรรมจำลอง และรูปทางด้านขวาเป็นด้านล่างของรถเกษตรกรรมจำลอง ซึ่งประกอบไปด้วยมอเตอร์แบบ wiper จำนวนสองตัว ประกอบเข้ากับโครงเหล็ก และล้อ เพื่อใช้กำหนดทิศทางการเคลื่อนที่ของรถทางเกษตรกรรม จากนั้นทำการวางสถานีจลน์ และอุปกรณ์ที่เกี่ยวข้องลงบนรถทางเกษตรกรรมจำลอง ดังแสดงในรูปที่ 3.23



รูปที่ 3.23 การจัดวางสถานีจลน์ลงบนรถเกษตรกรรมจำลอง

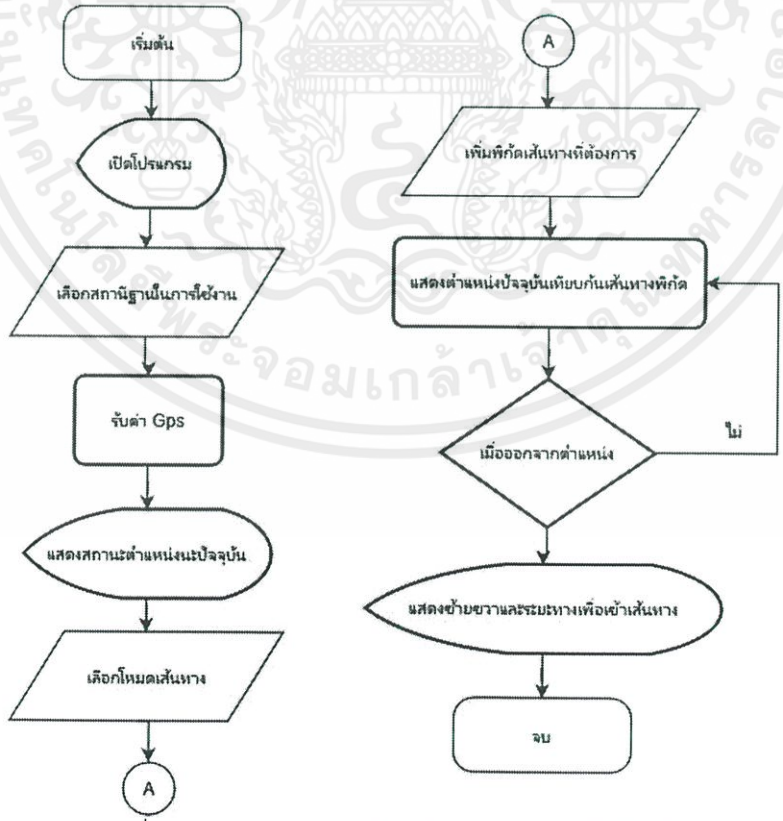
3.1.4 ออกแบบส่วนต่อประสานกราฟิกกับผู้ใช้

ทำการออกแบบส่วนต่อประสารกราฟิกกับผู้ใช้ เพื่อความสะดวกต่อผู้ใช้งาน ทำให้ผู้ใช้สามารถกำหนดเส้นทางการเดินรถทางเกษตรกรรม และแสดงผลค่าสถานะของรถทางการเกษตรผ่านส่วนต่อประสานกราฟิกกับผู้ใช้ โดยมีส่วนประกอบหลัก 3 หน้าต่างการใช้งาน ได้แก่ หน้าการตั้งค่าการทำงานของระบบ หน้าแสดงค่าการเคลื่อนที่ของสถานีจลน์ และหน้าแสดงพิกัดเส้นทางของสถานีจลน์ การออกแบบส่วนต่อประสานกราฟิกกับผู้ใช้ สามารถแสดงรูปแบบการทำงานได้ดังรูปที่ 3.24



รูปที่ 3.24 การออกแบบส่วนต่อประสานกราฟิกกับผู้ใช้

ผังงานการทำงานของส่วนต่อประสานกราฟิกกับผู้ใช้แสดงได้ดังรูปที่ 3.25



รูปที่ 3.25 การทำงานของส่วนต่อประสานกราฟิกกับผู้ใช้

จากรูปที่ 3.25 การทำงานของโปรแกรมเริ่มจากหน้าเลือกสถานีฐานในการใช้งานโดยการเลือกสถานีฐานจะมีหลายประเภทให้เลือก เช่น ใช้สถานีฐานบนตึก 12 ชั้น เป็นต้น

หน้าแสดงสถานะตำแหน่งปัจจุบันจะแสดงตำแหน่งปัจจุบันของสถานีจลน์ออกมาเป็นละติจูด ลองจิจูด และ เวลาวันที่ เพื่อให้ผู้ใช้นั้นทราบตำแหน่งของสถานีจลน์

หน้าเลือกโหมดเส้นทางในการเดินรถทางการเกษตรนั้นจำเป็นต้องเลือกโหมดในการเพิ่มเส้นทางพิกัดเข้าไป โดยจะมีให้เลือกทั้งหมดสองโหมด Single waypoint และ Multi waypoint ซึ่ง Single waypoint นั้นคือการเพิ่มเส้นทางที่ล่ะจุดพิกัดตามลำดับเพื่อเชื่อมเส้นเข้าด้วยกัน ส่วน Multiway point เป็นการเพิ่มพิกัด 4 จุดพิกัดจากนั้นให้โปรแกรมคำนวนจุดพิกัดที่เหลือตามจำนวนเส้นที่ผู้ใช้ใส่ไป

หน้าเพิ่มพิกัดเส้นทางที่ต้องการในการเดินรถทางการเกษตรจะแสดงตำแหน่งพิกัดในปัจจุบันและเมื่อกดปุ่มเพิ่มเส้นทางพิกัดจะทำการเพิ่มพิกัดปัจจุบันเพื่อเพิ่มเข้าไปในเส้นทางพิกัด

หน้าแสดงระยะห่างระหว่างสถานีจลน์กับเส้นทางพิกัด จะแสดงระยะห่างระหว่างสถานีจลน์และพิกัดเส้นทางเมื่อสถานีจลน์ออกนอกเส้นทางและจะแสดงทิศทางที่สถานีจลน์จำเป็นต้องใช้ในการกลับเข้าเส้นทางพิกัด

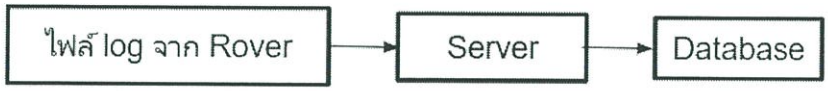
3.1.5 การเก็บข้อมูลการใช้งานลงบนฐานข้อมูล

ในการใช้งานสถานีจลน์ทุกครั้ง จะมีการเก็บค่าพิกัดตำแหน่งจากสายอากาศ ด้วยเทคนิค RTK พิกัดตำแหน่งของเส้นทาง ตลอดจนสถานะการทำงาน ณ เวลาที่ทำการใช้งาน ซึ่งจะทำการจัดเก็บเป็นไฟล์ .log ดังรูปที่ 3.26

```
2019/03/17,11:40:27,13.72738663,100.77634243,-4.978,0.0,49.327,left
2019/03/17,11:40:28,13.72737794,100.77634049,-4.401,0.0,50.289,left
2019/03/17,11:40:29,13.72736791,100.77633797,-3.753,0.0,51.399,left
2019/03/17,11:40:29,13.72736323,100.77633692,-3.691,0.0,51.917,left
2019/03/17,11:40:30,13.72735612,100.77633533,-3.261,0.0,52.704,left
2019/03/17,11:40:31,13.72735430,100.77633497,-3.436,0.0,52.905,left
2019/03/17,11:40:32,13.72735567,100.77633509,-3.922,0.0,52.753,left
2019/03/17,11:40:33,13.72735901,100.77633567,-4.467,0.0,52.383,left
2019/03/17,11:40:33,13.72735480,100.77633426,-4.238,0.0,52.848,left
```

รูปที่ 3.26 ไฟล์ .log ที่เก็บมาแบบเวลาจริง

จากรูปที่ 3.26 เมื่อเก็บไฟล์ .log ที่มีการบันทึก วัน เวลา ละติจูด ลองจิจูด ค่าความสูง รวมถึงค่าอัตราส่วนความแม่นยำของเทคนิค RTK ตลอดจนระยะห่างระหว่างค่าพิกัดตำแหน่งกับเส้นทาง และทิศทางที่ต้องเข้าไปยังเส้นทาง แล้วจะทำการส่งข้อมูลนี้ไปยังฐานข้อมูล โดยมีแผนผังการทำงาน ดังรูปที่ 3.26



รูปที่ 3.27 แผนผังการจัดเก็บข้อมูลจากสถานีจลน์ไปยังฐานข้อมูล

จากรูปที่ 3.27 หลังจากที่สถานีจลน์เก็บข้อมูลเป็นไฟล์ .log แบบเวลาจริง แล้วสิ้นสุดการทำงานของส่วนต่อประสานกราฟิกกับผู้ใช้ จะมีการส่งค่าไฟล์ .log นี้ ไปยังเซิฟเวอร์ของภาควิชาโทรคมนาคม ที่มีไอพีแอดเดรส 161.246.18.205 และทำการอัพโหลดขึ้นไปบนตารางของฐานข้อมูล ดังรูปที่ 3.28

date	time	latitude	longitude	height	ratio	distance	side
2019-03-22	11:42:48	13.727511405944824	100.7763442993164	-30.914	0.0	0.000	
2019-03-22	11:42:49	13.727461814880371	100.77632141113281	-26.29	0.0	0.000	
2019-03-22	11:42:51	13.72742748260498	100.77630615234375	-23.995	0.0	0.519	turn_left
2019-03-22	11:42:52	13.727400779724121	100.77629852294922	-21.35	0.0	0.593	turn_left
2019-03-22	11:42:53	13.727372169494629	100.77629852294922	-15.539	0.0	0.042	correct
2019-03-22	11:42:54	13.727351188659668	100.77629089355469	-10.954	0.0	0.000	
2019-03-22	11:42:55	13.727336883544922	100.77629089355469	-7.849	0.0	0.739	turn_right
2019-03-22	11:42:56	13.727341651916504	100.77629089355469	-7.581	0.0	0.873	turn_right
2019-03-22	11:42:57	13.727357864379883	100.77629852294922	-8.321	0.0	1.014	turn_right
2019-03-22	11:42:58	13.727364540100098	100.77630615234375	-8.609	0.0	1.041	turn_right
2019-03-22	11:42:58	13.727370262145996	100.77630615234375	-8.333	0.0	1.162	turn_right
2019-03-22	11:42:59	13.727372169494629	100.77630615234375	-8.421	0.0	1.171	turn_right
2019-03-22	11:43:00	13.727371215820312	100.77630615234375	-8.937	0.0	1.045	turn_right
2019-03-22	11:43:01	13.72735595703125	100.77629852294922	-7.131	0.0	1.092	turn_right
2019-03-22	11:43:02	13.727343559265137	100.77629852294922	-5.647	0.0	1.124	turn_right
2019-03-22	11:43:03	13.72733211517334	100.77629089355469	-4.29	0.0	1.144	turn_right
2019-03-22	11:43:03	13.72732162475586	100.77629089355469	-3.096	0.0	1.167	turn_right
2019-03-22	11:43:04	13.727289199829102	100.77628326416016	0.633	0.0	1.265	turn_right
2019-03-22	11:43:05	13.72728443145752	100.77628326416016	1.185	0.0	1.282	turn_right
2019-03-22	11:43:06	13.727261543273926	100.77627563476562	3.706	0.0	1.349	turn_right
2019-03-22	11:43:07	13.72726058959961	100.77627563476562	3.781	0.0	0.006	correct
2019-03-22	11:43:08	13.72726821899414	100.77627563476562	2.355	0.0	0.039	correct
2019-03-22	11:43:09	13.727278709411621	100.77627563476562	0.765	0.0	0.099	correct
2019-03-22	11:43:10	13.727287292480469	100.77628326416016	-0.177	0.0	0.000	
2019-03-22	11:43:12	13.727314949035645	100.77628326416016	-4.862	0.0	0.539	turn_right

รูปที่ 3.28 ตารางฐานข้อมูลบนเซิฟเวอร์

จากรูปที่ 3.28 เมื่อทำการอัพโหลดข้อมูลเข้าไปยังตาราง โดยจะสามารถดึงค่าในแต่ละหลักออกมาใช้ประมวลผล และแสดงบนหน้าเว็บไซต์ได้

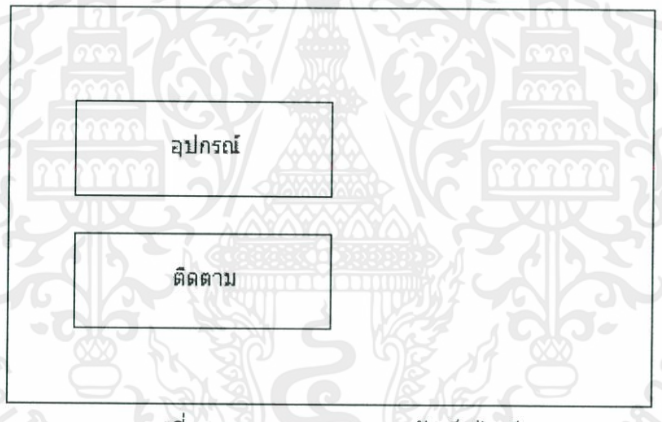
3.1.6 การออกแบบหน้าเว็บไซต์

ทำการออกแบบหน้าเว็ปไซต์เพื่อติดตาม และตรวจสอบระบบนำร่องทางการเกษตร ซึ่งประกอบด้วยหน้าแสดงอุปกรณ์ โดยการติดตามนั้นต้องทำการเข้าสู่ระบบ เพื่อเข้าถึงข้อมูลภายในฐานข้อมูล และสามารถตรวจสอบข้อมูลภายในฐานข้อมูลได้

3.1.6.1 หน้าแรกของเว็บไซต์

ออกแบบให้ผู้ใช้สามารถใช้งานได้ง่าย โดยกำหนดให้มีประเภทการใช้งานสองประเภท ได้แก่

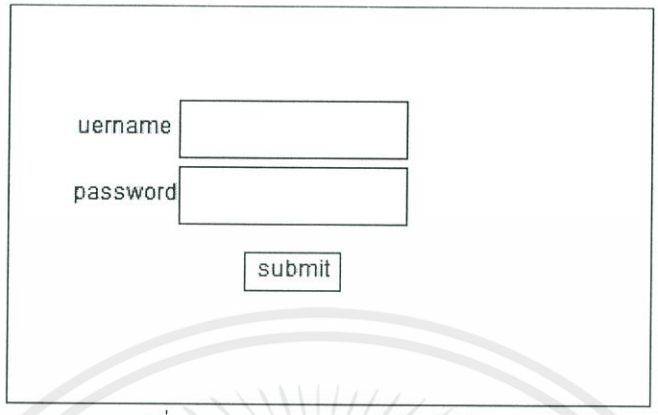
- หน้าต่างแสดงอุปกรณ์ที่ใช้งาน และติดตามทั้งหมด เพื่อตรวจสอบการทำงานของสถานีจลน์ โดยมีรูปแบบหน้าเว็บไซต์ดังรูปที่ 3.29



รูปที่ 3.29 การออกแบบหน้าเว็บไซต์

3.1.6.2 หน้าเข้าสู่ระบบ

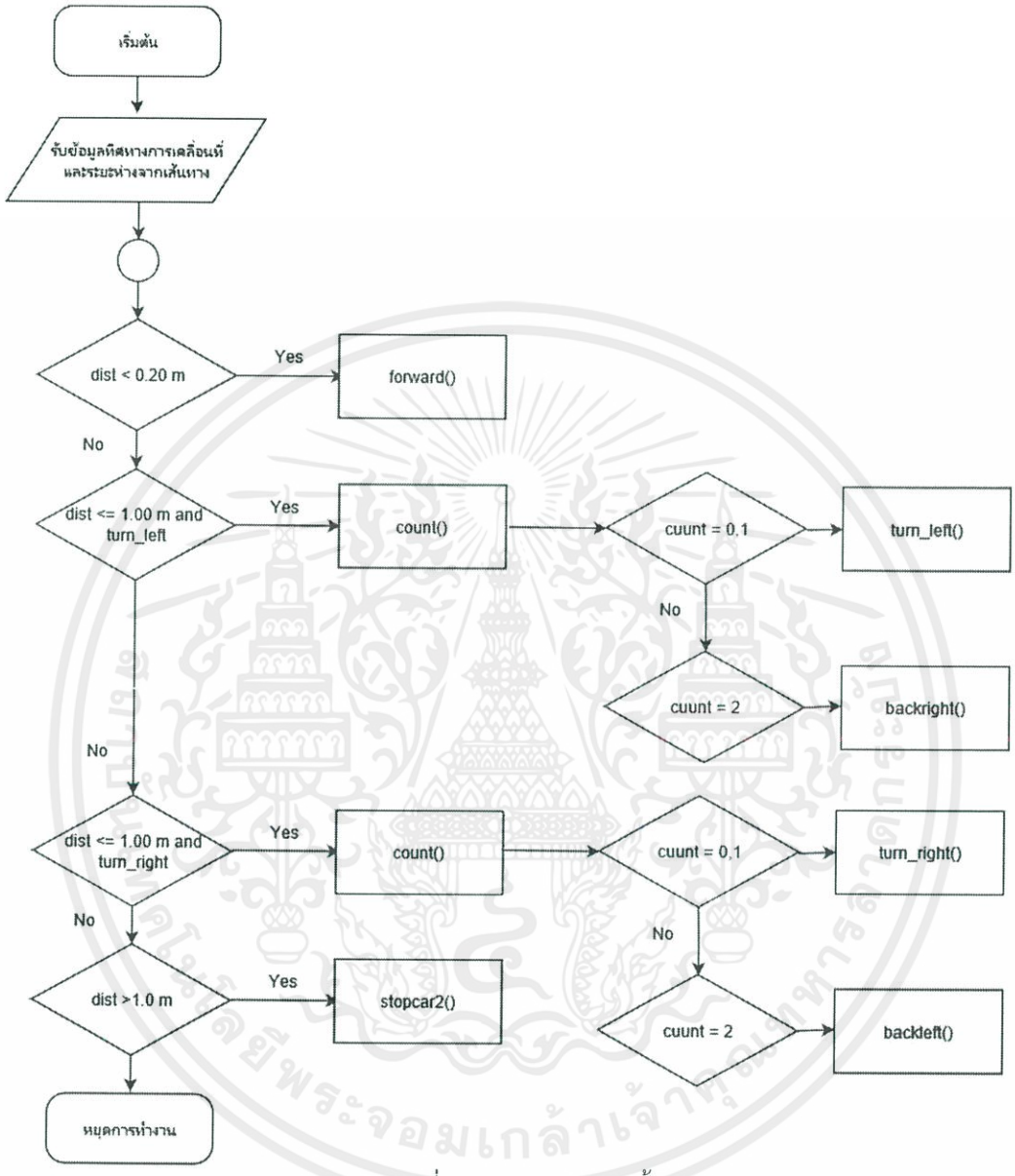
ออกแบบให้มีแทบสำหรับกรอกข้อมูลเพื่อเข้าสู่ระบบ ซึ่งกำหนดให้มีชื่อผู้ใช้และรหัสผ่าน เมื่อกรอกข้อมูลแล้วจะมีปุ่มให้กดเข้าสู่ระบบ หลังจากกดปุ่มจะนำไปสู่หน้าแสดงผลข้อมูลของสถานีจลน์ โดยแสดงการออกแบบหน้าเข้าสู่ระบบดังรูปที่ 3.30



รูปที่ 3.30 การออกแบบหน้าเข้าสู่ระบบ

3.1.7 การออกแบบระบบบังคับเลี้ยวอัตโนมัติ

ในการออกแบบระบบบังคับเลี้ยวอัตโนมัติ จำเป็นต้องปรับความเร็วในการหมุนล้อรถทางเกษตรกรรมจำลอง ให้มีความเร็วที่เหมาะสมต่อการใช้งาน โดยควบคุมจากโปรแกรมควบคุมความเร็วการหมุนของล้อรถ ซึ่งมีลักษณะโครงสร้างของระบบบังคับเลี้ยวอัตโนมัติแสดงดังรูปที่ 3.31



รูปที่ 3.31 ระบบบังคับเลี้ยวอัตโนมัติ

จากรูปที่ 3.31 แสดงระบบบังคับเลี้ยวอัตโนมัติ โดยเริ่มจากการรับข้อมูลทิศทางการเคลื่อนที่ (turn_left or turn_right) และระยะห่างจากเส้นทาง (dist) โดยแบ่งการตัดสินใจเป็น 4 รูปแบบ ได้แก่

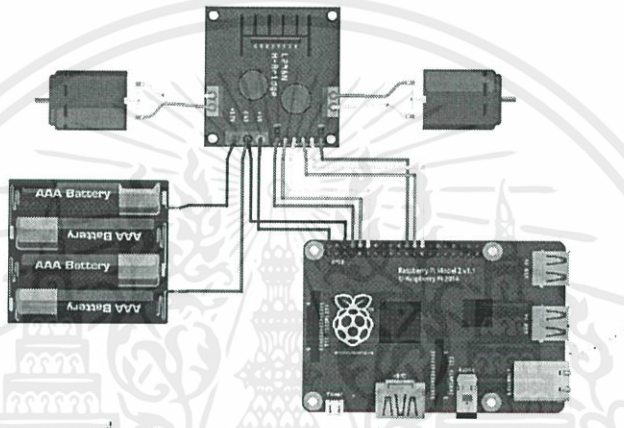
- dist < 0.20 m หมายถึง รถอยู่ในเส้นทางให้สั่งใช้งานคำสั่ง forward()

- dist <= 1.00 m turn_left หมายถึง รถอยู่ทางด้านขวาของเส้นทาง ให้ทำการเลี้ยวซ้าย โดยสั่งใช้งานคำสั่ง turn_left() 2 ครั้ง และคำสั่ง backright() 1 ครั้ง

- dist <= 1.00 m turn_right หมายถึง รถอยู่ทางด้านซ้ายของเส้นทางให้ทำการเลี้ยวขวา โดยสั่งใช้งานคำสั่ง turn_right() 2 ครั้ง และคำสั่ง backleft() 1 ครั้ง

- dist > 1 m หมายถึง รถอยู่นอกเส้นทางให้ทำการสั่งใช้งานคำสั่ง stopcar2()

เมื่อทำการเขียนโปรแกรมเสร็จแล้ว ทำการเชื่อมต่อ Raspberry pi ไอซี L298N และมอเตอร์เข้าด้วยกัน ดังแสดงในรูปที่ 3.32



รูปที่ 3.32 การเชื่อมต่อ Raspberry Pi เข้าไอซี L298N และมอเตอร์

จากรูป 3.32 แสดงการเชื่อมต่อระหว่าง Raspberry pi ไอซี L298N และมอเตอร์ โดยไอซี L298N จะรับค่าโลจิกอินพุตจาก Raspberry pi เพื่อใช้สำหรับควบคุมความเร็ว และทิศทางในการหมุนของมอเตอร์ รวมถึงรับค่าแรงดันไฟฟ้าจากแบตเตอรี่ เพื่อใช้ในการขับเคลื่อนมอเตอร์ โดยสามารถขับเคลื่อนมอเตอร์ได้สูงสุดจำนวน 2 ตัว

3.2 เครื่องมือที่ใช้ในการทดลอง

โปรแกรม RTKLIB	
โปรแกรมไพธอน	
Raspberry Pi	1 ชุด
เซ็นเซอร์ รุ่น GY-85	1 ชุด
สาย Micro USB	2 สาย
สายนำสัญญาณ SMA	1 สาย
U-Blox NEO-M8T Receiver	1 ชุด
TW-3740	1 ชุด
โทรศัพท์มือถือแอนดรอยด์	1 ชุด

จอแสดงผลขนาด 7 นิ้ว	1 ชุด
IC L298N	1 ตัว
รถเกษตรกรรมจำลอง	1 ชุด
แบตเตอรี่ 12V6Ah	1 ชุด

3.3 การจัดเก็บผลการทดลอง

3.3.1 การออกแบบสถานีจลน์ (Rover)

3.3.1.1 ทดสอบรับข้อมูลพิกัด มาประมวลผลเพื่อทำ RTK

ก่อนออกทดสอบภาคสนาม ได้มีการทดสอบรับข้อมูลพิกัด มาประมวลผลเพื่อทำ RTK แบบตั้งอยู่กับที่ประมาณ 30 นาที เพื่อทดสอบว่าสถานีจลน์ที่สร้างขึ้นสามารถนำมาคำนวณตำแหน่งด้วยเทคนิค RTK ได้

3.3.1.2 ส่วนการปรับแต่งพารามิเตอร์ใน Configuration file สำหรับโปรแกรม RTKRCV

ในส่วนนี้ จะทำการทดสอบคลิกเลือกค่าพารามิเตอร์ที่ต้องการปรับแก้ลงในหน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้ในส่วนการ Configuration จากนั้นจึงทำการตรวจสอบค่าพารามิเตอร์ที่ถูกปรับแก้ภายในไฟล์ Config ว่าตรงกับที่ป้อนค่าผ่านหน้าต่างส่วนต่อประสานกราฟิกกับผู้ใช้หรือไม่ สุดท้ายจึงนำไฟล์ Config ที่ได้ไปทดสอบการรับค่าพิกัดตำแหน่งด้วยโปรแกรม RTKRCV โดยการทดสอบในส่วนนี้สามารถแสดงขั้นตอนในรูปแบบไดอะแกรมดังรูปที่ 3.33

รูปที่ 3.33 ขั้นตอนการทดสอบหน้าต่างการตั้งค่า Configuration สำหรับโปรแกรม RTKRCV

3.3.2 การออกแบบระบบบังคับเลี้ยวอัตโนมัติ

ในส่วนนี้จะทำการทดสอบ โดยสร้างเส้นทางสำหรับเดินรถทางเกษตรกรรม จากนั้นนำรถทางเกษตรกรรมจำลองมาทดลองขับเคลื่อนบนเส้นทาง โดยมีการแทรกแซงจากผู้ทดสอบ ให้เกิดรูปแบบการทำงานในทุกรูปแบบ ซึ่งจะมีรูปแบบทั้งหมด 4 รูปแบบ ดังนี้

- รถอยู่บนเส้นทาง (รถอยู่ห่างจากเส้นทางไม่เกิน 20 เซนติเมตร)
- รถอยู่ทางด้านซ้ายของเส้นทาง (รถอยู่ห่างจากเส้นทาง 20 เซนติเมตรถึง 1 เมตร)
- รถอยู่ทางด้านขวาของเส้นทาง (รถอยู่ห่างจากเส้นทาง 20 เซนติเมตรถึง 1 เมตร)
- รถอยู่ห่างจากเส้นทางมากเกินไป จำเป็นต้องหยุดการทำงาน (รถอยู่ห่างจากเส้นทางมากกว่า 1 เมตร)

3.3.3 การทดสอบความแม่นยำของการระบุพิกัดตำแหน่ง

ในส่วนนี้จะทำการทดสอบ โดยสร้างเส้นทางการเดินรถบนดาดฟ้าอาคาร 12 ชั้น คณะวิศวกรรมศาสตร์ และแปลงเกษตร บริเวณอาคารเรียน ECC คณะวิศวกรรมศาสตร์ เพื่อเปรียบเทียบประสิทธิภาพการทำงานของสถานีจลน์ในบริเวณที่แตกต่างกัน และมีสิ่งกีดขวางสัญญาณจีเอ็นเอสเอส โดยแบ่งการทดสอบออกเป็นสองลักษณะ ดังนี้

- การทดสอบแบบอุดมคติ

เป็นการทดสอบบริเวณดาดฟ้า อาคาร 12 ชั้น ซึ่งเป็นบริเวณที่ตั้งของสถานีฐาน ส่งผลให้มีค่าความแม่นยำใกล้เคียงกับความเป็นจริงมากที่สุด โดยการทดสอบนี้จะใช้รถเข็นเพื่อจำลองเป็นรถทางเกษตรกรรม

- การทดสอบในแปลงเกษตร

เป็นการทดสอบบริเวณแปลงเกษตร ด้านหลังอาคาร ECC ซึ่งการทดสอบนี้จะมีสิ่งกีดขวาง และระยะห่างระหว่างสถานีฐานกับสถานีจลน์ โดยการทดสอบนี้จะใช้รถเกษตรกรรมจริง เพื่อทดสอบความแม่นยำในการใช้งานจริง เมื่อการทดสอบเสร็จสิ้น จะมีการเก็บรวมรวมข้อมูล และประมวลผลค่าความผิดพลาดของพิกัดตำแหน่ง สำหรับใช้เปรียบเทียบประสิทธิภาพการทำงานของสถานีจลน์

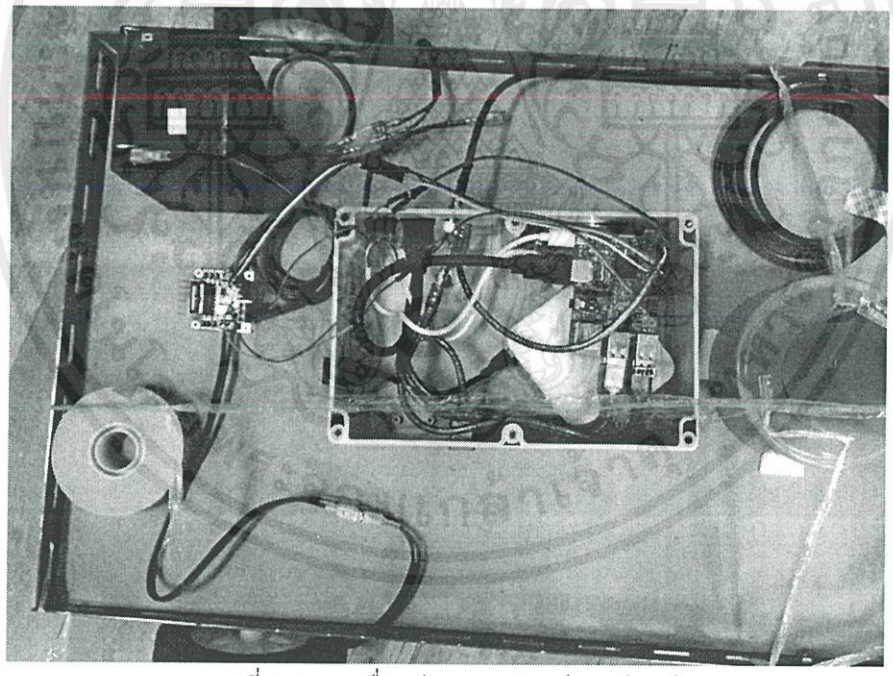
บทที่ 4

ผลการทดลอง

4.1 ผลการทดสอบอุปกรณ์สถานีจลน์

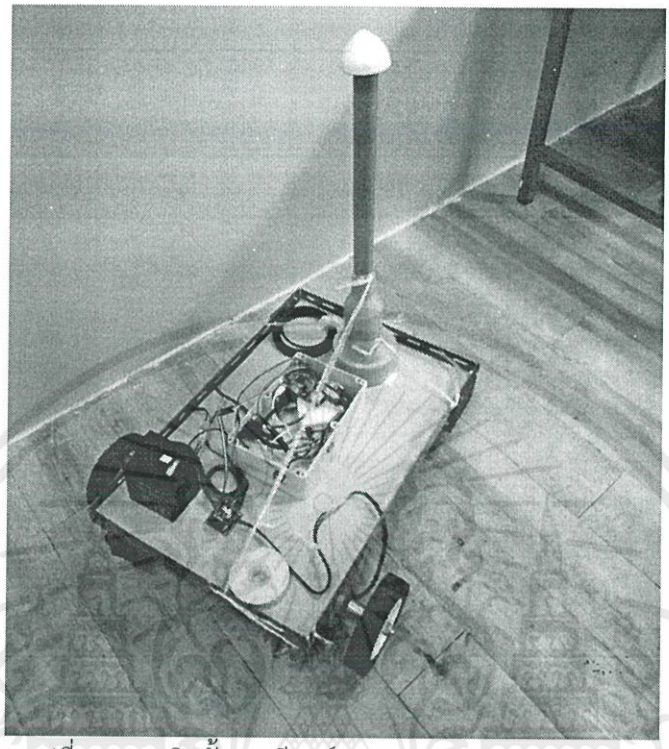
4.1.1 ทดสอบ RTK จากเครื่องรับจีพีเอสที่อ้างอิงจากสถานีฐาน

ทีมงานได้ทำการต่ออุปกรณ์เพื่อประกอบสถานีจลน์ มีขั้นตอนการดำเนินงาน โดยเริ่มจากการต่อ Raspberry Pi เข้ากับ U-Blox NEO-M8T สายอากาศ TW-3740 จากนั้นต่อเข้ากับ Power Bank ขนาด 5 โวลต์ 20 แอมป์ชั่วโมง แล้วทำการเชื่อมต่อเข้าเครือข่าย Mobile Hotspot หรือโมดูล 3G ที่ติดตั้งไว้ เพื่อรับค่าแก้จากเครื่องรับสัญญาณจีเอ็นเอสเอสสถานีฐานชนิดสองความถี่ที่อาคาร 12 ชั้น คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง มาประมวลผลเพื่อแก้ไขพิกัดตำแหน่งโดยใช้เทคนิค RTK ดังรูปที่ 4.1



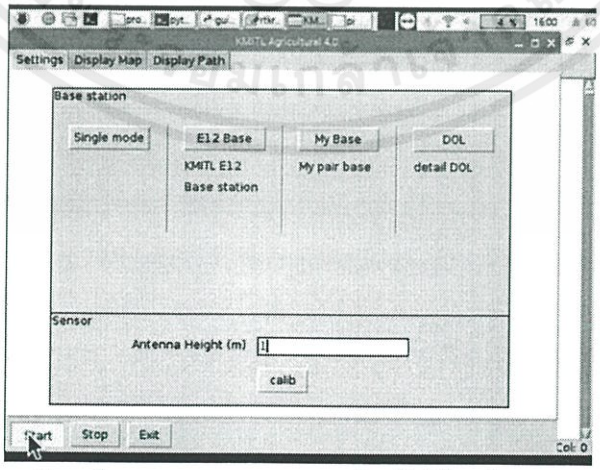
รูปที่ 4.1 การเชื่อมต่อของอุปกรณ์สถานีจลน์

จากรูปที่ 4.1 แสดงการติดตั้งอุปกรณ์สถานีจลน์บนรถเข็น โดยเปรียบการเคลื่อนที่ของรถเป็นการเดินรถทางเกษตรกรรม แล้วทำการจัดวางอุปกรณ์บนรถจำลองทางเกษตรกรรมดังแสดงในรูปที่ 4.2



รูปที่ 4.2 การติดตั้งสถานีจลน์บนรถทางเกษตรกรรมจำลอง

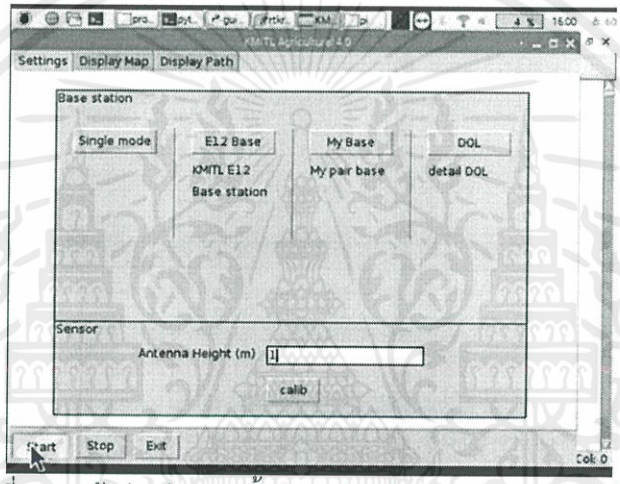
จากรูปที่ 4.2 แสดงการติดตั้งสถานีจลน์บนรถทางเกษตรกรรมจำลองที่มีความสูง 1 เมตรจากล้อรถเข็น ณ ดาดฟ้าอาคาร 12 ชั้น คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าฯ เจ้าคุณทหารลาดกระบัง จากนั้นเปิดการทำงานของโปรแกรม KMITL Agricultural 4.0 แล้วทำการตั้งค่าพารามิเตอร์ความสูงของสายอากาศ ดังรูปที่ 4.3



รูปที่ 4.3 การตั้งค่าพารามิเตอร์ความสูงของสายอากาศสำหรับใช้งานโปรแกรม KMITL Agricultural 4.0

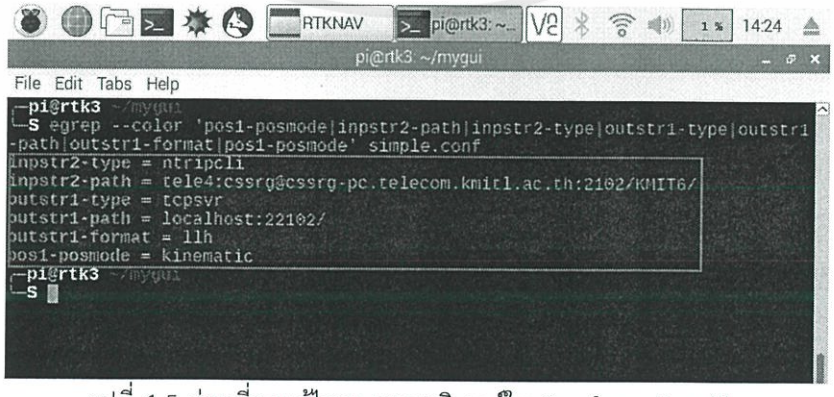
จากรูปที่ 4.3 เป็นการตั้งค่าพารามิเตอร์ความสูงของสายอากาศสำหรับใช้งานโปรแกรม KMITL Agricultural 4.0 จากนั้นทำการกดปุ่ม calib เพื่อตั้งค่าอ้างอิงเริ่มต้นให้กับเซ็นเซอร์ สำหรับแก้ไขค่าพิกัดความเอียงของสายอากาศขณะที่ต่อเซ็นเซอร์

4.1.2 ผลการทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้ส่วนการปรับแต่งพารามิเตอร์ใน Configuration file สำหรับโปรแกรม RTKRCV



รูปที่ 4.4 หน้าต่างส่วนการตั้งค่าบนส่วนต่อประสานกราฟิกกับผู้ใช้

จากรูปที่ 4.4 แสดงหน้าต่างของโปรแกรมที่เขียนขึ้นเพื่อปรับแต่งพารามิเตอร์สำหรับโปรแกรม RTKRCV โดยเมื่อเรียกใช้งานจะมีการปรับแต่งสถานะให้เลือกจำนวน 4 รูปแบบ ได้แก่ Single mode E12 base My base และ DOL ซึ่งการปรับแต่งสถานะจะมีการแก้ไข Configuration file ดังแสดงในรูปที่ 4.5

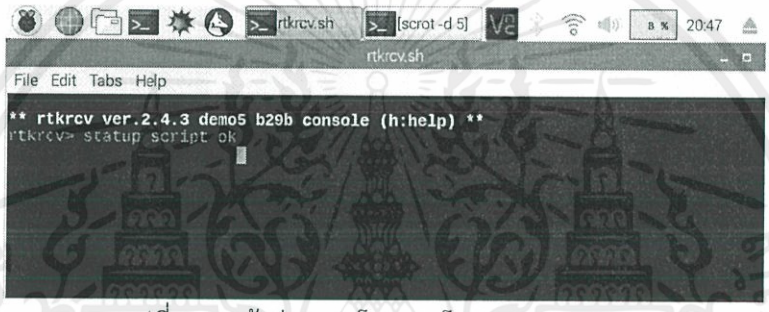


รูปที่ 4.5 ส่วนที่ถูกแก้ไขของพารามิเตอร์ใน Configuration file

4.1.3 ผลการทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้ส่วนการนำทาง

การทดสอบการทำงานของส่วนต่อประสานกราฟิกกับผู้ใช้ เป็นการทดสอบการทำงานของโปรแกรม RTKRCV ณ บริเวณชั้นบนสุดของตึก 12 ชั้น คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าฯ เจ้าคุณทหาร ลาดกระบัง

เริ่มต้นจากเรียกใช้โปรแกรม RTKRCV บนหน้าจอแสดงผลของอุปกรณ์สถานีจลน์ เพื่อคำนวณตำแหน่งด้วยเทคนิค RTK แล้ว เมื่อสั่งใช้งานโปรแกรมสำเร็จ จะแสดงหน้าต่างคอนโซลของโปรแกรม RTKRCV ดังรูปที่ 4.6

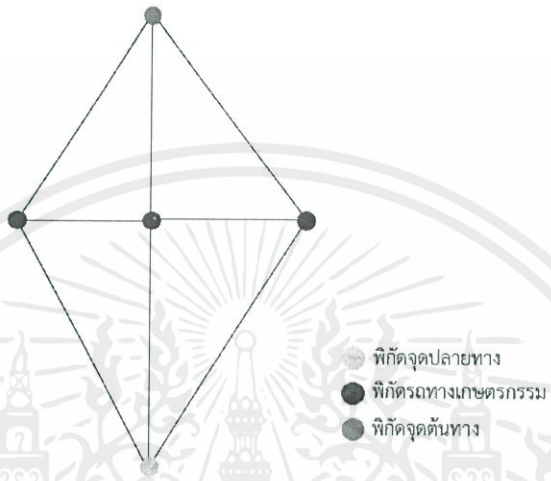


รูปที่ 4.6 หน้าต่างคอนโซลของโปรแกรม RTKRCV

จากนั้นทำการป้อนพิกัดเส้นทางจากหน้าต่างเพิ่มเส้นทาง โดยการเพิ่มพิกัดเส้นทางนั้นมีอยู่ 2 แบบ ได้แก่ การเพิ่มพิกัดเส้นทางแบบทีละจุด และการเพิ่มพิกัดเส้นทางแบบกำหนดจุดอ้างอิง 4 จุด แล้วป้อนคำสั่งให้ระบบสร้างเส้นทางระหว่างจุดอ้างอิงขึ้นมาตามจำนวนที่ผู้ใช้ต้องการ โดยในการทดลองเป็นการเพิ่มจุดพิกัดเส้นทางแบบทีละจุด ซึ่งจะได้เส้นทางดังรูปที่ 4.7

รูปที่ 4.7 จุดต้นทาง และจุดปลายทางในการทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้

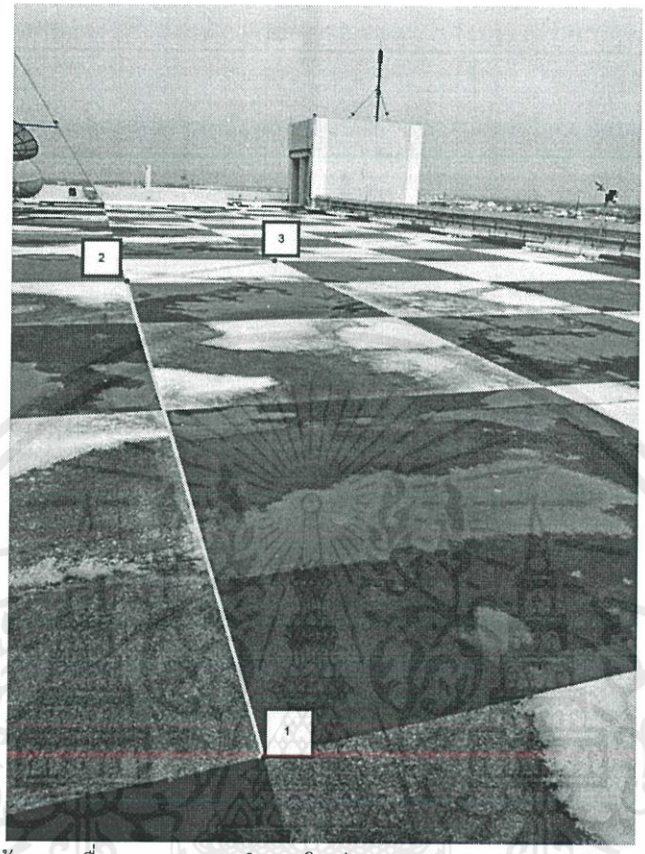
ซึ่งในการทดสอบในสถานที่จริง ได้ทำการทดสอบจำนวน 3 ครั้ง และมีการทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้ ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 การทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้

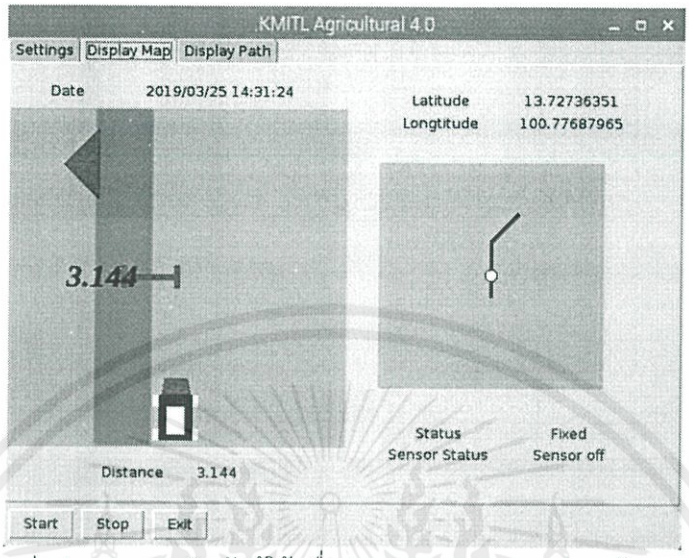
จากรูปที่ 4.8 แสดงการทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้ โดยนำค่าพิกัดจากอุปกรณ์สถานีจลน์มาคำนวณ เพื่อพิจารณาว่าตำแหน่งปัจจุบันของสถานีจลน์ว่าอยู่บริเวณด้านขวาด้านซ้าย หรืออยู่บนเส้นทาง ตลอดจนแสดงพิกัดปลายทาง และมุมจากพิกัดต้นทาง เพื่อให้ผู้ใช้งานตรวจสอบความถูกต้องได้ด้วยโปรแกรม MATLAB

4.1.3.1 การทดสอบส่วนต่อประสานกราฟิกกับผู้ใช้ส่วนการพิจารณาตำแหน่งของรถทางเกษตรกรรมเปรียบเทียบกับเส้นทาง



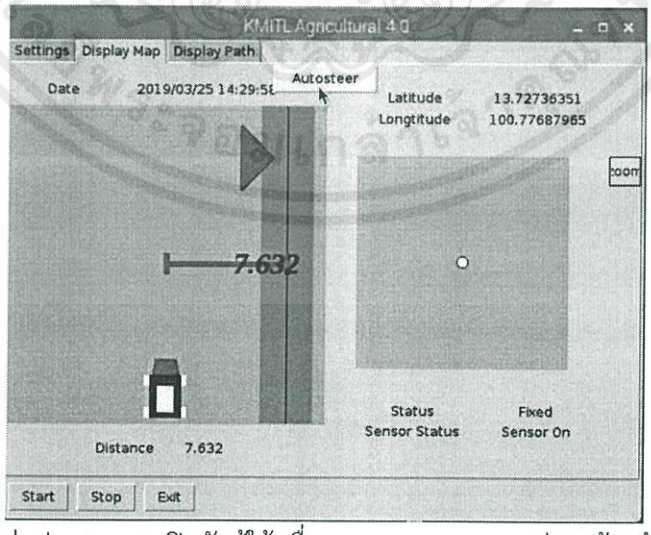
รูปที่ 4.9 การสร้างเส้นทางเพื่อทดสอบการทำงานในส่วนการพิจารณาตำแหน่งของรถทางเกษตรกรรมเปรียบเทียบกับเส้นทาง

จากรูปที่ 4.9 แสดงพิกัดเส้นทางที่ทำการจัดเก็บจำนวน 3 จุด เพื่อสร้างเส้นทางสำหรับทดสอบการทำงานของส่วนต่อประสานกราฟิกกับผู้ใช้ ในส่วนการพิจารณาตำแหน่งของรถทางเกษตรกรรมเปรียบเทียบกับเส้นทาง โดยผลการทดสอบแสดงดังรูปที่ 4.10 ถึงรูปที่ 4.12



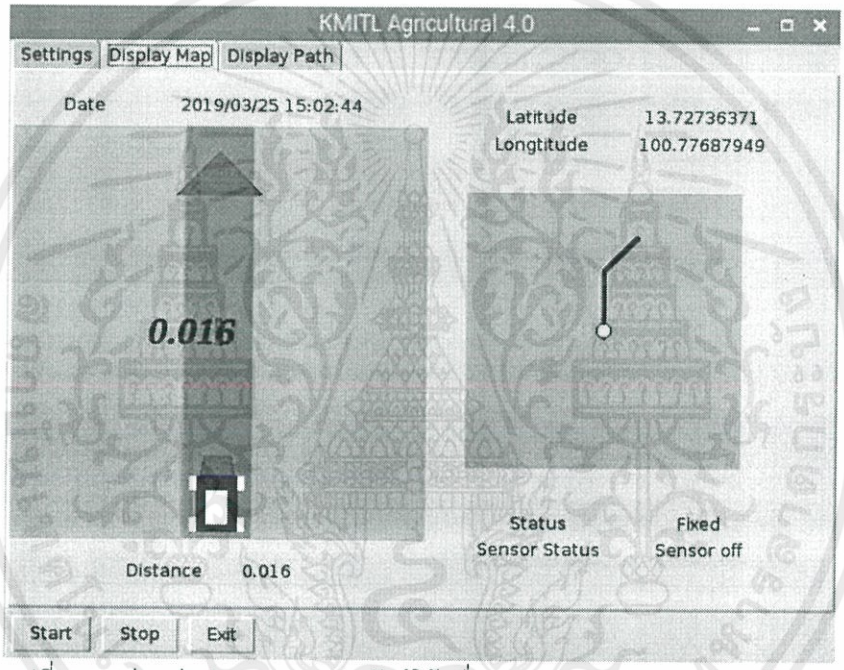
รูปที่ 4.10 ส่วนต่อประสานกราฟิกกับผู้ใช้ เมื่อรถทางเกษตรกรรมอยู่ทางด้านขวาของเส้นทาง

จากรูปที่ 4.10 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ เมื่อรถทางเกษตรกรรมอยู่ทางด้านขวาของเส้นทาง โดยระบบจะแสดงแผนที่ขนาดเล็กที่บริเวณด้านขวาของหน้าต่างโปรแกรม เพื่อบอกตำแหน่งของรถทางเกษตรกรรมเปรียบเทียบกับเส้นทาง และแสดงทิศทางที่ต้องเคลื่อนที่เพื่อกลับเข้าสู่เส้นทางการเดินรถผ่านลูกศรสีแดงที่อยู่บริเวณด้านซ้ายของหน้าต่างโปรแกรม โดยในการทดสอบพบว่ารถทางเกษตรกรรมมีค่าพิกัดอยู่ที่ 13.72736351N 100.77687965E และมีระยะห่างกับเส้นทาง 3.144 เมตร



รูปที่ 4.11 ส่วนต่อประสานกราฟิกกับผู้ใช้ เมื่อรถทางเกษตรกรรมอยู่ทางด้านซ้ายของเส้นทาง

จากรูปที่ 4.11 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ เมื่อรถทางเกษตรกรรมอยู่ทางด้านซ้ายของเส้นทาง โดยระบบจะแสดงแผนที่ขนาดเล็กที่บริเวณด้านขวาของหน้าต่างโปรแกรม เพื่อบอกตำแหน่งของรถทางเกษตรกรรมเปรียบเทียบกับเส้นทาง และแสดงทิศทางที่ต้องเคลื่อนที่เพื่อกลับเข้าสู่เส้นทางการเดินรถ ผ่านลูกศรสีแดงที่อยู่บริเวณด้านซ้ายของหน้าต่างโปรแกรม โดยในการทดสอบพบว่ารถทางเกษตรกรรมมีค่าพิกัดอยู่ที่ 13.72736351N 100.77687965E และมีระยะห่างกับเส้นทาง 7.632 เมตร



รูปที่ 4.12 ส่วนต่อประสานกราฟิกกับผู้ใช้ เมื่อรถทางเกษตรกรรมอยู่บนเส้นทาง

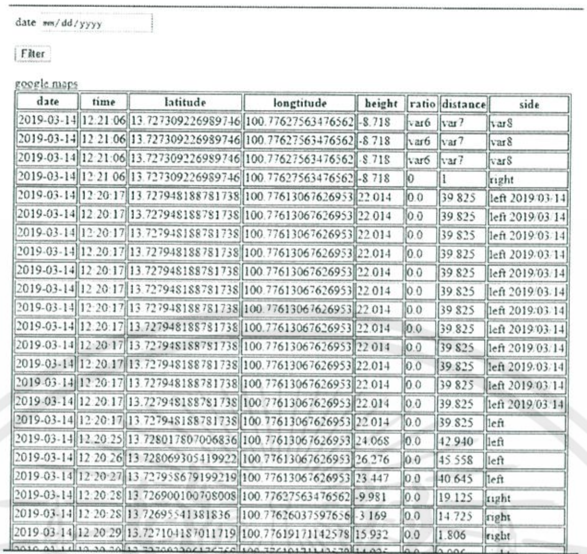
จากรูปที่ 4.12 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ เมื่อรถทางเกษตรกรรมอยู่บนเส้นทาง โดยระบบจะแสดงแผนที่ขนาดเล็กที่บริเวณด้านขวาของหน้าต่างโปรแกรม เพื่อบอกตำแหน่งของรถทางเกษตรกรรมเปรียบเทียบกับเส้นทาง และแสดงทิศทางที่ต้องใช้การเดินรถ ผ่านลูกศรสีแดงที่อยู่บริเวณด้านซ้ายของหน้าต่างโปรแกรม โดยในการทดสอบพบว่ารถทางเกษตรกรรมมีค่าพิกัดอยู่ที่ 13.72736371N 100.77687949E และมีระยะห่างกับเส้นทาง 0.016 เมตร ซึ่งมีขนาดน้อยมาก จึงสามารถอนุมานได้ว่ารถทางเกษตรกรรมอยู่บนเส้นทางการเดินรถ

4.1.3 ส่วนแสดงการติดตามรถทางเกษตรกรรมผ่านทางหน้าเว็ปไซต์

หลังจากที่สถานีจลน์ได้ทำการส่งค่าตัวแปรที่ใช้บอกสถานะของการทำงานไปบนฐานข้อมูล จะสามารถดูสถานการณ์ทำงานย้อนหลังได้ที่เว็ปไซต์ ดังแสดงในรูปที่ 4.13

รูปที่ 4.13 หน้าเว็ปไซต์หน้าแรก
(http://train.telecom.kmitl.ac.th/rtktrack/)

จากรูปที่ 4.13 การติดตามสถานะของสถานีจลน์ย้อนหลังนั้น ทำได้โดยการเลือกลักษณะการใช้งานแบบติดตาม และเข้าสู่ระบบทำให้ผู้ใช้สามารถเข้าถึงประวัติและสถานะย้อนหลังของสถานีจลน์ ดังรูปที่ 4.14



date mm/dd/yyyy

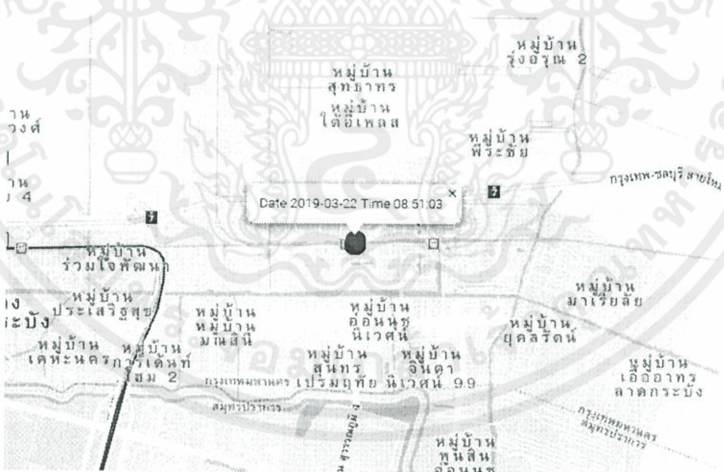
Filter

google maps

date	time	latitude	longtitude	height	ratio	distance	side
2019-03-14	12:21:06	13.727309226989746	100.77627563476562	-8.718	var6	var7	var8
2019-03-14	12:21:06	13.727309226989746	100.77627563476562	-8.718	var6	var7	var8
2019-03-14	12:21:06	13.727309226989746	100.77627563476562	-8.718	var6	var7	var8
2019-03-14	12:21:06	13.727309226989746	100.77627563476562	-8.718	0	1	right
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left 2019/03/14
2019-03-14	12:20:17	13.727948188781738	100.77613067626953	22.014	0.0	39.825	left
2019-03-14	12:20:25	13.728017807006836	100.77613067626953	24.068	0.0	42.940	left
2019-03-14	12:20:26	13.728069305419922	100.77613067626953	26.276	0.0	45.558	left
2019-03-14	12:20:27	13.727958679199219	100.77613067626953	23.447	0.0	40.645	left
2019-03-14	12:20:28	13.726900100708008	100.77627563476562	-9.981	0.0	19.125	right
2019-03-14	12:20:28	13.72695541381836	100.77626037597656	-3.169	0.0	14.725	right
2019-03-14	12:20:29	13.727104187011719	100.77619171142578	15.932	0.0	1.806	right

รูปที่ 4.14 หน้าเว็บไซต์หน้าแสดงข้อมูล

จากรูปที่ 4.14 ให้ทำการป้อนวันที่ที่ต้องการติดตามสถานะย้อนหลังของสถานีจลน์ จะแสดงผลแผนที่ และสถานะดังรูปที่ 4.15



รูปที่ 4.15 หน้าเว็บไซต์หน้าแสดงข้อมูลบนแผนที่

จากรูปที่ 4.15 ในการติดตามสถานะย้อนหลังผู้ใช้สามารถทราบตำแหน่งของสถานีจลน์และวันที่สถานีจลน์ทำงาน

4.2 ผลการทดสอบการอ่านค่ามุมที่แสดงการเคลื่อนที่ของวัตถุโดยใช้ IMU Sensor รุ่น GY-85

การตรวจสอบความแม่นยำในการวัดมุม Roll Pitch และ Yaw จะใช้การบอกร้อยละความแตกต่างระหว่างค่าที่กำหนดกับค่าที่วัดได้ สามารถคำนวณได้จากสมการที่ 4.1

$$\text{ร้อยละความแตกต่างในการวัดมุม} = \left| \frac{(b-a)}{a} \right| \times 100 \qquad (4.1)$$

โดยที่ a = มุมที่กำหนดในหน่วยองศา และ b = มุมที่อุปกรณ์ Sensor รุ่น GY-85 วัดได้ในหน่วยองศา

ผลการเปรียบเทียบค่าของมุม Roll ตั้งแต่ -15 ถึง 15 องศา (ยกเว้นมุม 0 องศา) โดยมีการเปลี่ยนแปลงค่ามุมครั้งละ 5 องศา แสดงผลดังตารางที่ 4.1

ตารางที่ 4.1 ผลการเปรียบเทียบการวัดมุม Roll ด้วยเซนเซอร์ GY-85

ค่า Roll ที่กำหนด	ค่า Roll ที่วัดได้จริง	ร้อยละความแตกต่างในการวัดมุม
-15	-15.16	1.06
-10	-10.07	0.70
-5	-5.05	1.00
0	0.03	-
5	5.18	3.54
10	10.1	1.00
15	15.17	1.13

จากตารางที่ 4.1 ผลจากการทดสอบพบว่ามีค่าความคลาดเคลื่อนอยู่ในระดับที่ต่ำ โดยมีค่าความคลาดเคลื่อนอยู่ในช่วง 0.70 ถึง 3.54 ยกเว้นบริเวณที่ 0 องศา เนื่องจากส่วนของสมการเศษส่วนมีค่าเป็น 0 จึงไม่สามารถนิยามค่าความแตกต่างของมุมได้ตามหลักคณิตศาสตร์

ผลการวัดมุม Pitch ด้วยเซนเซอร์ GY-85 จากมุม -15 องศา ถึงมุม 15 องศา โดยเปลี่ยนมุมครั้งละ 5 องศา แสดงผลดังตารางที่ 4.2

ตารางที่ 4.2 ผลการเปรียบเทียบการวัดมุม Pitch ด้วยเซนเซอร์ GY-85

ค่า Pitch ที่กำหนด	ค่า Pitch ที่วัดได้จริง	ร้อยละความแตกต่างในการวัดมุม
-15	-15.16	1.06
-10	-10.06	0.60

-5	-4.95	1.01
0	0.06	-
5	5.19	3.73
10	10.06	0.60
15	14.86	0.94

จากตารางที่ 4.2 ผลจากการทดสอบพบว่ามีค่าความคลาดเคลื่อนอยู่ในระดับที่ต่ำ โดยมีค่าความคลาดเคลื่อนอยู่ในช่วง 0.60 ถึง 3.73 ยกเว้นมนบริเวณที่ 0 องศา เนื่องจากส่วนของสมการเศษส่วนมีค่าเป็น 0 จึงไม่สามารถนิยามค่าความแตกต่างของมุมได้ตามหลักคณิตศาสตร์

ผลการวัดมุม Yaw ด้วยเซนเซอร์ GY-85 จะเริ่มวัดจากมุม 0 องศา ถึงมุม 350 องศา โดยเปลี่ยนมุมไปครั้งละ 10 องศา มีผลการวัดดังตารางที่ 4.3

ตารางที่ 4.3 ผลการวัดมุม Yaw ด้วยเซนเซอร์ GY-85

ค่า Yaw ที่กำหนด	ค่า Yaw ที่วัดได้จริง	ร้อยละความแตกต่างในการวัด
0	0	0.00
10	10	0.00
20	20	0.00
30	30	0.00
40	41	2.40
50	51	1.98
60	61	1.65
70	71	1.42
80	80	0.00
90	91	1.11
100	101	0.10
110	111	-.90
120	120	0.00
130	131	0.76
140	140	0.00
150	151	0.66
160	161	0.62

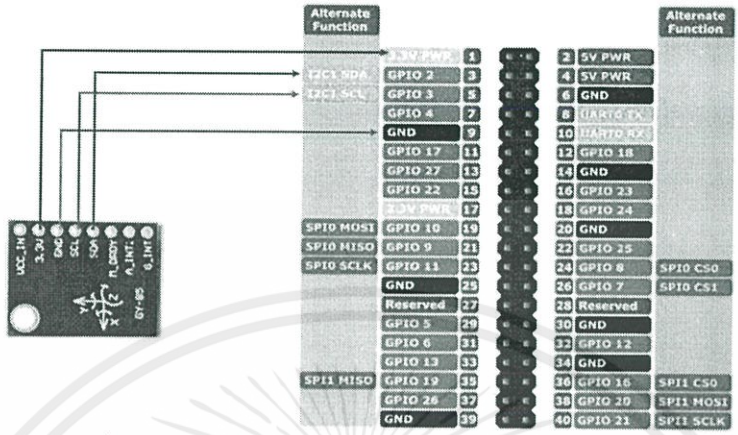
170	171	0.58
180	180	0.00
190	190	0.00
200	201	0.50
210	211	0.48
220	220	0.00
230	232	0.87
240	242	0.42
250	252	0.79
260	262	0.76
270	272	0.73
280	283	1.07
290	292	0.69
300	302	0.66
310	312	0.64
320	323	0.93
330	332	0.60
340	342	0.59
350	351	0.29

ผลการวัดมุม Yaw จากตารางที่ 4.3 พบว่ามีค่าร้อยละความแตกต่างมีค่าไม่เกินร้อยละ 2.40 หรือมีความคลาดเคลื่อนไม่เกิน 2 องศา ซึ่งเพียงพอต่อการใช้งานกับรถทางเกษตรกรรม

4.3 ผลการทดสอบการเชื่อมต่อระหว่าง IMU Sensor รุ่น GY-85 กับ Raspberry Pi

4.4.1 การเขียนโปรแกรม และการทดสอบการใช้งานของโปรแกรมแสดงค่ามุม Roll Pitch และ Yaw

ทำการเชื่อมต่อโมดูล GY-85 เข้ากับ Raspberry Pi ดังรูปที่ 4.16 จากนั้นทำการเขียนโปรแกรม imu_kalman.py ดังรูปที่ 4.17 เพื่อทดสอบการทำงานของโมดูล GY-85 แสดงค่าของมุม (องศา) และเก็บค่าของมุม Roll Pitch และ Yaw ที่ได้ไว้ในไฟล์ Calib.txt

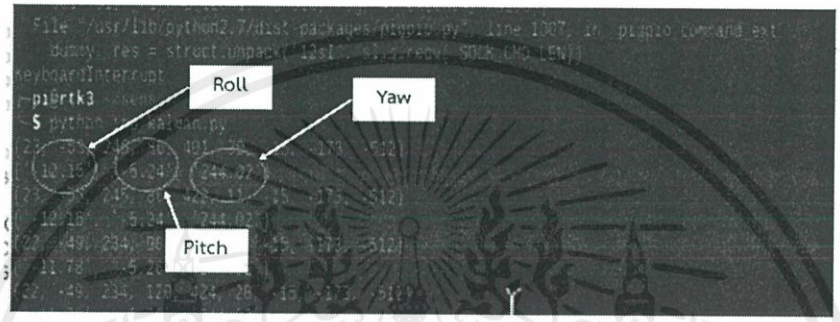


รูปที่ 4.16 การเชื่อมต่อโมดูล GY-85 เข้ากับ Raspberry Pi

จากรูปที่ 4.16 แสดงให้เห็นถึงการเชื่อมต่อโมดูล GY-85 เข้ากับ Raspberry Pi ผ่านขา GPIO แบบ Bus I2C โดยเชื่อมต่อขา 3.3 V และขา GND ของอุปกรณ์ทั้งสองเข้าด้วยกัน เพื่อจ่ายไฟเลี้ยงจาก Raspberry Pi ไปยังเซ็นเซอร์ GY-85 จากนั้นทำการเชื่อมต่อขา SDA และ SCL ของอุปกรณ์ทั้งสองเข้าด้วยกัน เพื่อส่งข้อมูลที่วัดได้ไปยัง Raspberry Pi สำหรับใช้ในการประมวลผลต่อไป

รูปที่ 4.17 โปรแกรม imu_kalman.py

รูปที่ 4.17 แสดงให้เห็นถึงรูปแบบคำสั่งของโปรแกรม imu_kalman.py ซึ่งเป็นการคำนวณค่ามุม Roll Pitch และ Yaw โดยใช้ค่าความเร่งที่วัดได้จากเครื่องวัดความเร่ง (Accelerometer) หลังจากนั้นทำการเปิดหน้าต่าง Terminal ในอุปกรณ์ Raspberry Pi เพื่อสั่งใช้งานโปรแกรม imu_kalman.py ดังรูปที่ 4.18



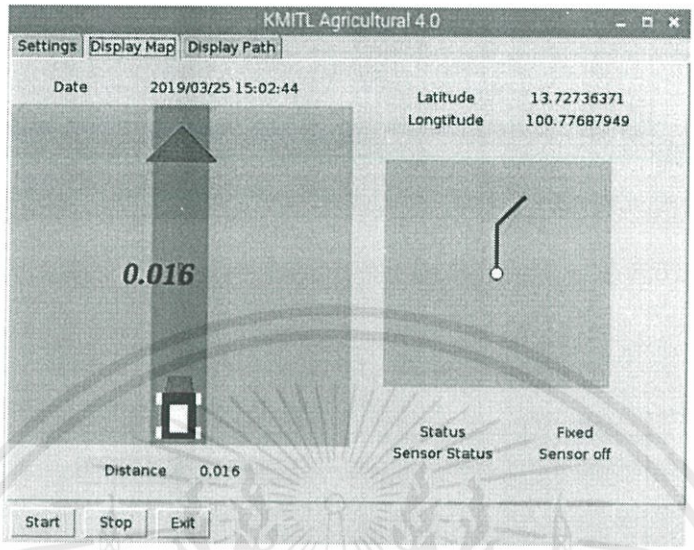
รูปที่ 4.18 ผลการสั่งใช้งานโปรแกรม imu_kalman.py ผ่านหน้าต่าง Terminal

จากรูปที่ 4.18 แสดงให้เห็นถึงค่ามุม Roll Pitch และ Yaw โดยจะแสดงแบบเวลาจริง และทำการเก็บข้อมูลไว้ในไฟล์ Calib.txt เพื่อให้ง่ายต่อการใช้งานสำหรับการเขียนโปรแกรมในส่วนถัดไป

4.4 ผลการทดสอบระบบบังคับเลี้ยวอัตโนมัติ

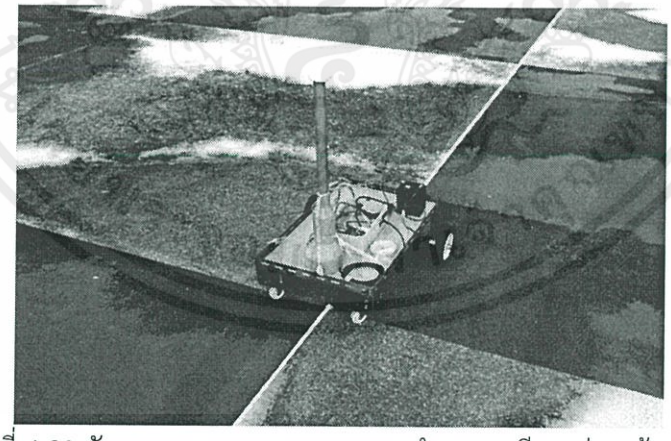
4.4.1 ระบบบังคับเลี้ยวอัตโนมัติกรณีรถอยู่บนเส้นทาง

จากการออกแบบการทำงานของระบบบังคับเลี้ยวอัตโนมัติ พบว่าลักษณะการทำงานในกรณีที่รถทางเกษตรกรรมจำลองอยู่บนเส้นทาง จะมีเงื่อนไขที่สำคัญสองส่วน ได้แก่ ระยะห่างจากเส้นทางเดินรถต้องไม่เกิน 20 เซนติเมตร และไม่มีค่าทิศทางของรถ ดังแสดงในรูปที่ 4.19



รูปที่ 4.19 ส่วนต่อประสานกาฟิกกับผู้ใช้กรณีรถทางเกษตรกรรมจำลองอยู่บนเส้นทาง

จากรูปที่ 4.19 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ กรณีรถทางเกษตรกรรมจำลองอยู่บนเส้นทาง พบว่ามีระยะห่างระหว่างรถทางเกษตรกรรมจำลองกับเส้นทาง 1.6 เซนติเมตร และมีทิศทางการเคลื่อนที่ไปทางด้านหน้า โดยพิจารณาจากลูกศรสีแดง เมื่อเงื่อนไขของกรณีรถอยู่บนเส้นทางครบแล้ว รถจะมีการเคลื่อนที่ไปทางด้านหน้า ดังแสดงในรูปที่ 4.20

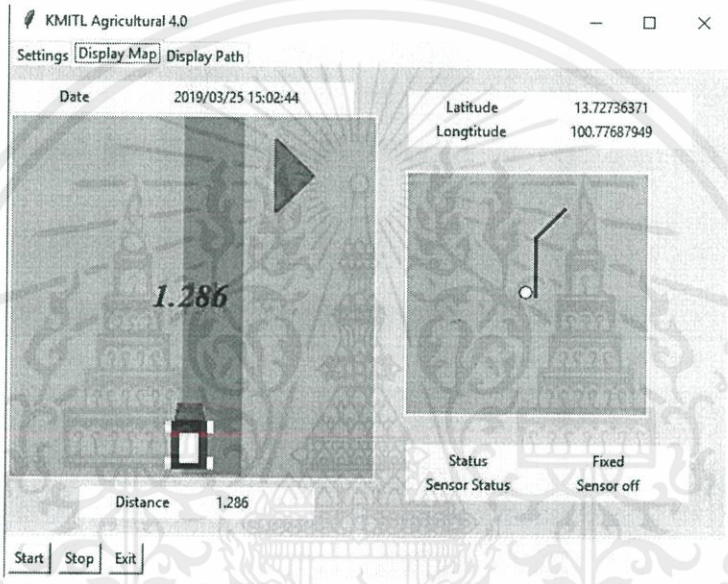


รูปที่ 4.20 ลักษณะของรถทางเกษตรกรรมจำลองกรณีรถอยู่บนเส้นทาง

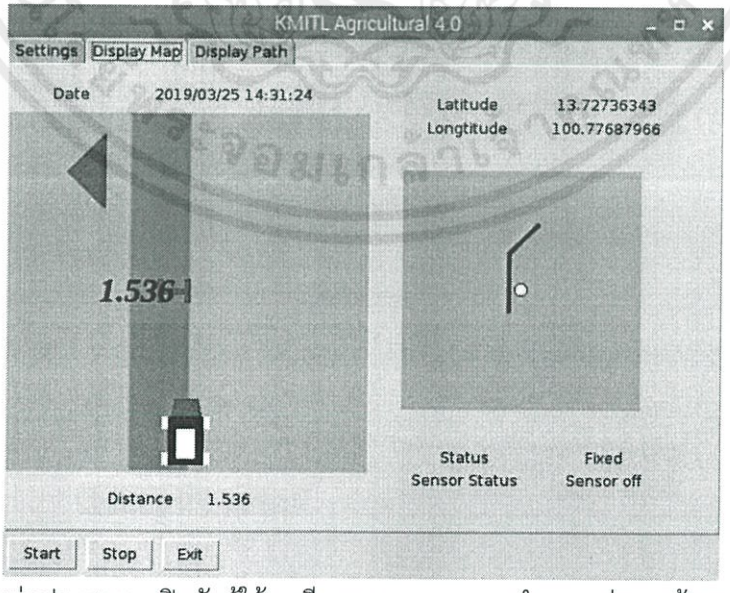
จากรูปที่ 4.20 แสดงการเคลื่อนที่ไปทางด้านหน้าของรถทางเกษตรกรรมจำลอง โดยรถทางเกษตรกรรมจำลองจะมีจำนวนรอบการหมุนของล้อขับเคลื่อนทางด้านซ้าย และขวาเท่ากัน ส่งผลให้รถเคลื่อนที่เป็นเส้นตรงด้วยอัตราเร็วคงที่

4.4.2 ระบบบังคับเลี้ยวอัตโนมัติกรณีรถอยู่นอกเส้นทาง

จากการออกแบบการทำงานของระบบบังคับเลี้ยวอัตโนมัติ พบว่าลักษณะการทำงานในกรณีที่รถทางเกษตรกรรมจำลองอยู่นอกเส้นทาง จะมีเงื่อนไขที่สำคัญสองส่วน ได้แก่ ระยะห่างจากเส้นทางเดินรถต้องมากกว่า 1 เมตร และค่าทิศทางของรถเป็นได้ทั้ง turn_left และ turn_right ดังแสดงในรูปที่ 4.21 และรูปที่ 4.22

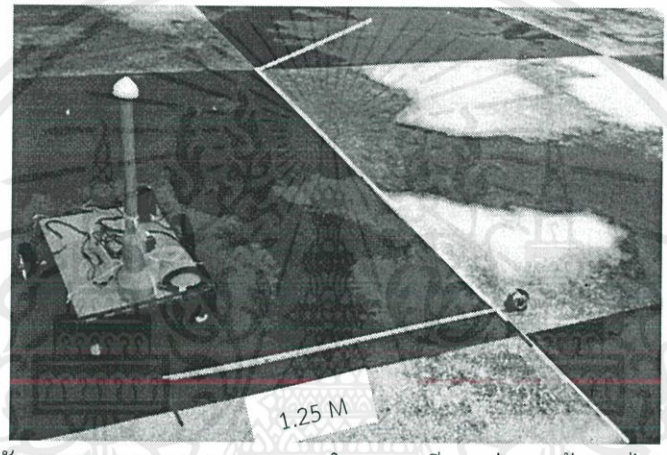


รูปที่ 4.21 ส่วนต่อประสานกาฟิกกับผู้ใช้กรณีรถทางเกษตรกรรมจำลองอยู่นอกเส้นทางทางด้านซ้าย

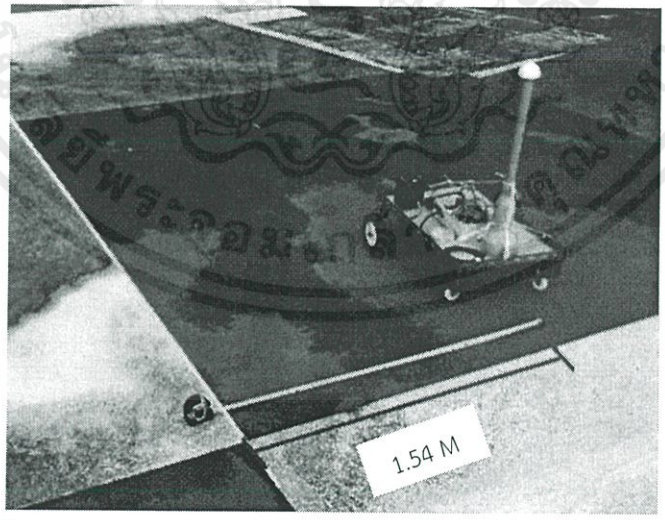


รูปที่ 4.22 ส่วนต่อประสานกาฟิกกับผู้ใช้กรณีรถทางเกษตรกรรมจำลองอยู่นอกเส้นทางทางด้านขวา

จากรูปที่ 4.21 และรูปที่ 4.22 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ กรณีรถทางเกษตรกรรมจำลองอยู่นอกเส้นทางไปทางด้านซ้าย และด้านขวา พบว่ามีระยะห่างระหว่างรถทางเกษตรกรรมจำลองกับเส้นทาง 1.286 เมตร และ 1.536 เมตร โดยมีทิศทางการเคลื่อนที่ไปทางด้านขวาในรูปที่ 4.21 และทางด้านซ้ายในรูปที่ 4.22 เมื่อเงื่อนไขของกรณีรถอยู่นอกเส้นทางครบแล้ว รถจะหยุดการเคลื่อนที่ ดังแสดงในรูปที่ 4.23 และรูปที่ 4.24



รูปที่ 4.23 ลักษณะของรถทางเกษตรกรรมจำลองกรณีรถอยู่นอกเส้นทางไปทางด้านซ้าย

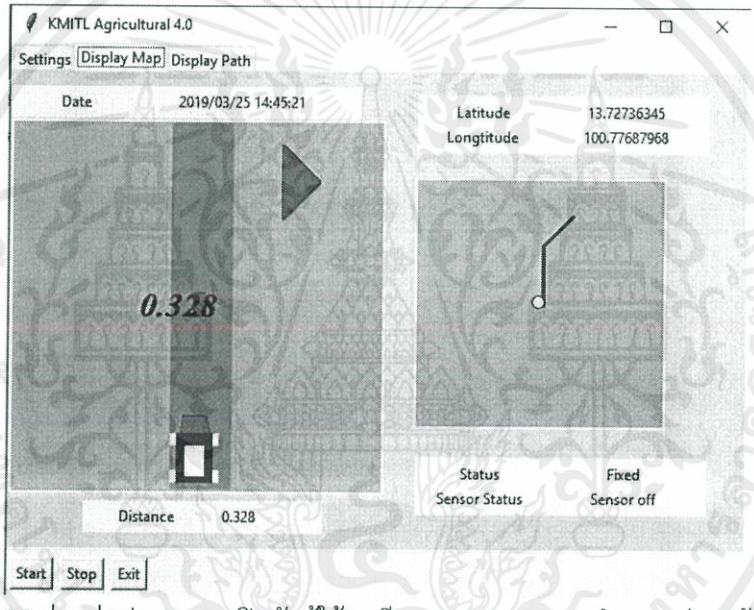


รูปที่ 4.24 ลักษณะของรถทางเกษตรกรรมจำลองกรณีรถอยู่นอกเส้นทางไปทางด้านขวา

จากรูปที่ 4.23 และรูปที่ 4.24 แสดงการหยุดการเคลื่อนที่ของรถทางเกษตรกรรมจำลอง โดยรถทางเกษตรกรรมจำลองจะหยุดการหมุนของล้อขับเคลื่อนทางด้านซ้าย และขวา ส่งผลให้รถหยุดเคลื่อนที่ และรอการประมวลผลรอบถัดไป

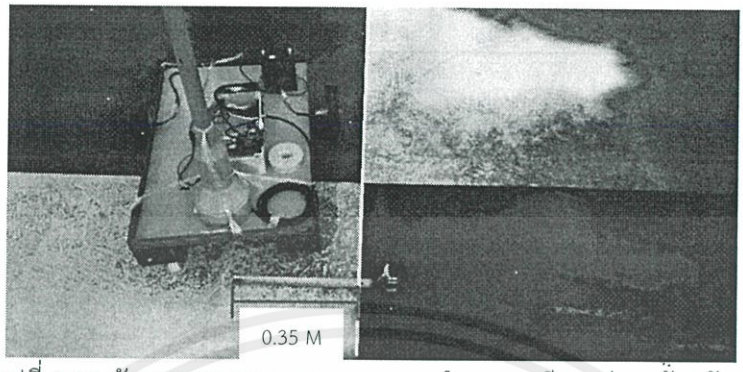
4.4.3 ระบบบังคับเลี้ยวอัตโนมัติกรณีรถอยู่ทางด้านซ้ายเส้นทาง

จากการออกแบบการทำงานของระบบบังคับเลี้ยวอัตโนมัติ พบว่าลักษณะการทำงานในกรณีที่รถทางเกษตรกรรมจำลองอยู่ทางด้านซ้ายเส้นทาง จะมีเงื่อนไขที่สำคัญสองส่วน ได้แก่ ระยะห่างจากเส้นทางเดินรถต้องมากกว่า 0.2 เมตร และค่าทิศทางของรถเป็น turn_right ดังแสดงในรูปที่ 4.25



รูปที่ 4.25 ส่วนต่อประสานกาฟิกกับผู้ใช้กรณีรถทางเกษตรกรรมจำลองอยู่ทางด้านซ้าย

จากรูปที่ 4.25 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ กรณีรถทางเกษตรกรรมจำลองอยู่ทางด้านซ้ายของเส้นทาง พบว่ามีระยะห่างระหว่างรถทางเกษตรกรรมจำลองกับเส้นทาง 0.328 เมตร และมีทิศทางการเคลื่อนที่ไปทางด้านขวา โดยพิจารณาจากลูกศรสีแดง เมื่อเงื่อนไขของกรณีรถอยู่ทางด้านซ้ายของเส้นทางครบแล้ว รถจะมีการเคลื่อนที่ไปทางด้านขวา ดังแสดงในรูปที่ 4.26

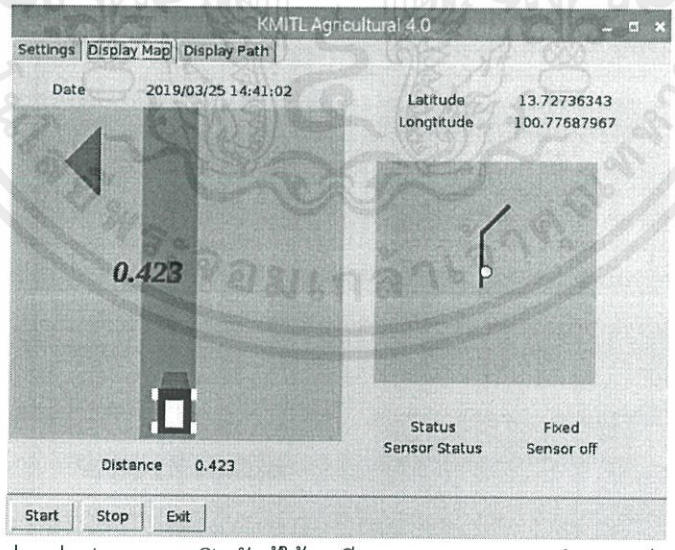


รูปที่ 4.26 ลักษณะของรถทางเกษตรกรรมจำลองกรณีรถอยู่ทางด้านซ้าย

จากรูปที่ 4.26 แสดงการเคลื่อนที่ไปทางด้านขวาของรถทางเกษตรกรรมจำลอง โดยรถทางเกษตรกรรมจำลองจะมีจำนวนรอบการหมุนของล้อขับเคลื่อนทางด้านซ้าย และขวาไม่เท่ากัน โดยล้อทางด้านซ้ายจะมีจำนวนรอบการหมุนมากกว่าทางด้านขวา ส่งผลให้รถเคลื่อนที่ไปทางด้านขวา

4.4.2 ระบบบังคับเลี้ยวอัตโนมัติกรณีรถอยู่ทางด้านขวาเส้นทาง

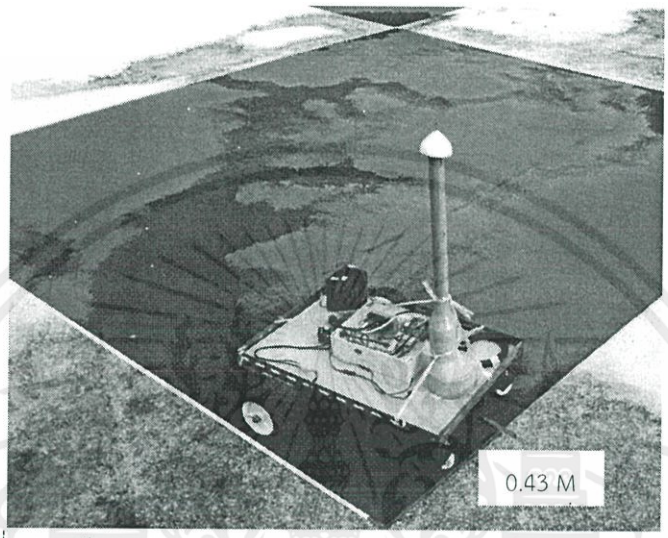
จากการออกแบบการทำงานของระบบบังคับเลี้ยวอัตโนมัติ พบว่าลักษณะการทำงานในกรณีที่รถทางเกษตรกรรมจำลองอยู่ทางด้านขวาเส้นทาง จะมีเงื่อนไขที่สำคัญสองส่วน ได้แก่ ระยะห่างจากเส้นทางเดินรถต้องมากกว่า 0.2 เมตร และค่าทิศทางของรถเป็น turn_left ดังแสดงในรูปที่ 4.27



รูปที่ 4.27 ส่วนต่อประสานกาฟิกกับผู้ใช้กรณีรถทางเกษตรกรรมจำลองอยู่ทางด้านขวา

จากรูปที่ 4.27 แสดงส่วนต่อประสานกราฟิกกับผู้ใช้ กรณีรถทางเกษตรกรรมจำลองอยู่ทางด้านขวาของเส้นทาง พบว่ามีระยะห่างระหว่างรถทางเกษตรกรรมจำลองกับเส้นทาง 0.423 เมตร

และมีทิศทางการเคลื่อนที่ไปทางด้านซ้าย โดยพิจารณาจากลูกศรสีแดง เมื่อเงื่อนไขของกรณีรถอยู่ทางด้านขวาของเส้นทางครบแล้ว รถจะมีการเคลื่อนที่ไปทางด้านซ้าย ดังแสดงในรูปที่ 4.28



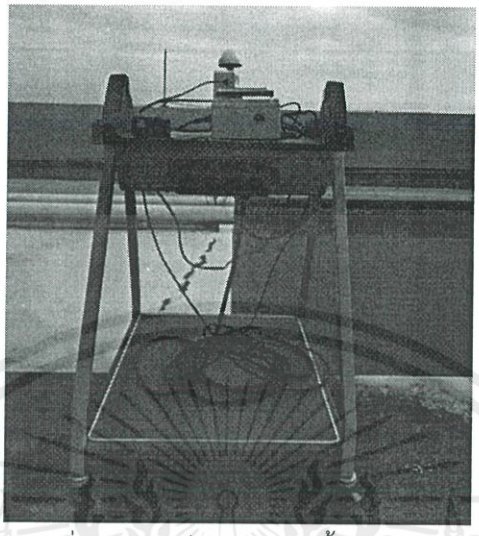
รูปที่ 4.28 ลักษณะของรถทางเกษตรกรรมจำลองกรณีรถอยู่ทางด้านขวา

จากรูปที่ 4.28 แสดงการเคลื่อนที่ไปทางด้านขวาของรถทางเกษตรกรรมจำลอง โดยรถทางเกษตรกรรมจำลองจะมีจำนวนรอบการหมุนของล้อขับเคลื่อนทางด้านซ้าย และขวาไม่เท่ากัน โดยล้อทางด้านซ้ายจะมีจำนวนรอบการหมุนน้อยกว่าทางด้านขวา ส่งผลให้รถเคลื่อนที่ไปทางด้านซ้าย

4.5 ผลการทดสอบประสิทธิภาพของสถานีจลน์

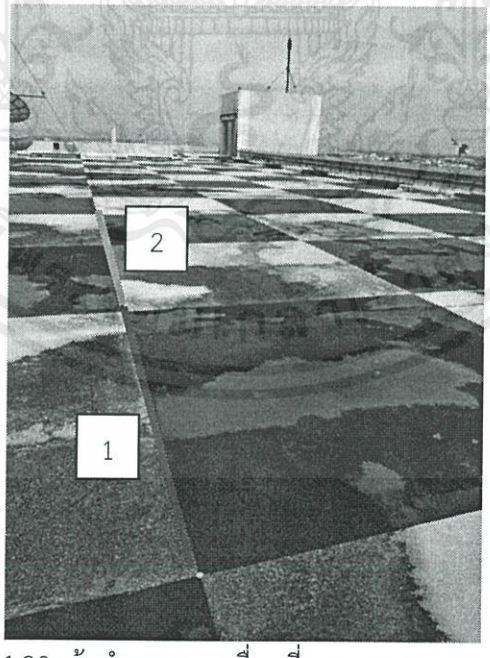
4.5.1 ผลการทดสอบแบบอุดมคติ

การทดสอบแบบอุดมคติ เริ่มจากการโดยจัดตั้งอุปกรณ์ลงบนรถเข็น ดังแสดงในรูปที่ 4.29



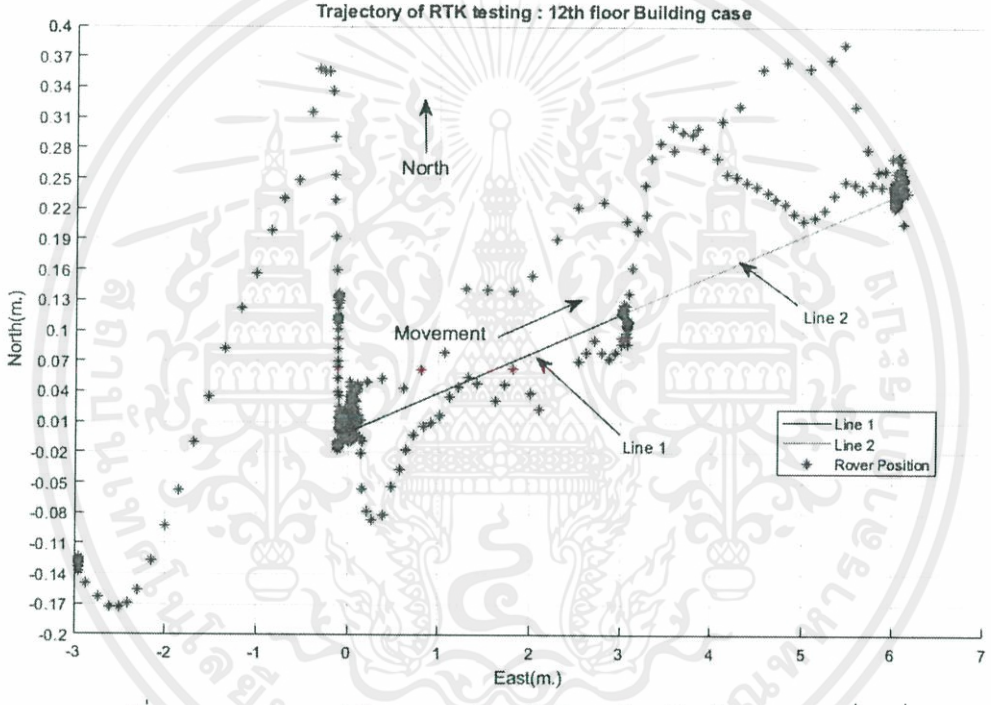
รูปที่ 4.29 รถเข็นสำหรับจัดตั้งสถานีจลน์

จากรูป 4.29 แสดงการติดตั้งสถานีจลน์ลงบนรถเข็น เพื่อทำการทดสอบประสิทธิภาพของสถานีจลน์ จากนั้นสร้างเส้นทางการเคลื่อนที่ ให้มีลักษณะแบบเดียวกับเส้นทางของรถทางเกษตรกรรม และให้เส้นทางการเคลื่อนที่อยู่ในแนวเดียวกับเส้นแบ่งแผ่นรองพื้น บนดาดฟ้าอาคาร 12 ชั้น ดังรูปที่ 4.30



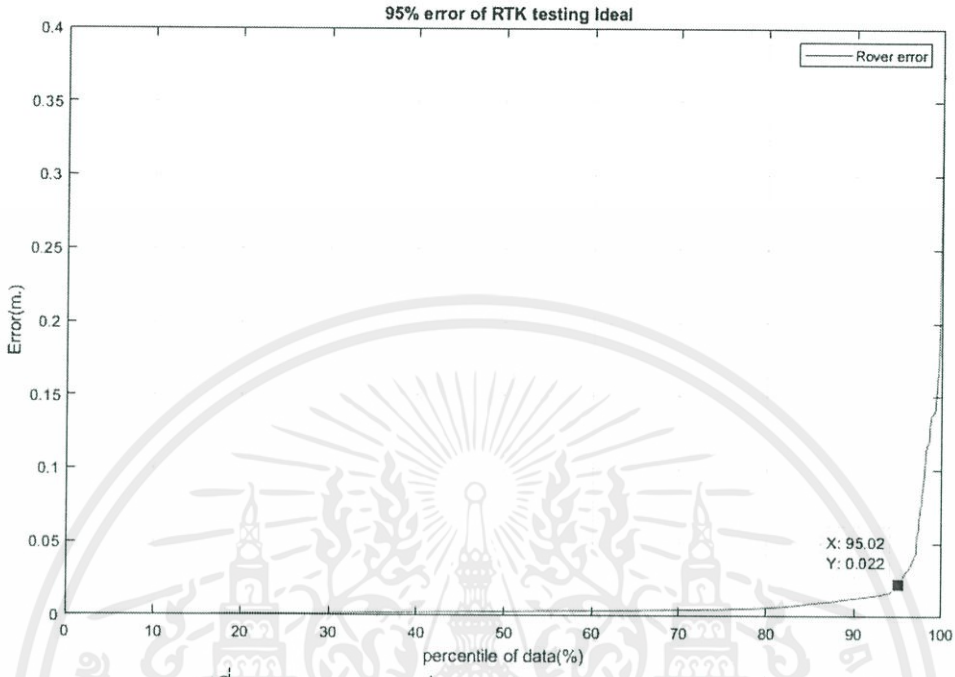
รูปที่ 4.30 เส้นจำลองการเคลื่อนที่ของรถทางเกษตรกรรม

จากรูปที่ 4.30 แสดงการสร้างเส้นจำลองการเคลื่อนที่ของรถทางเกษตรกรรม ประกอบด้วยเส้นทางสองเส้นทาง คือเส้นทางหมายเลข 1 และหมายเลข 2 จากนั้นทำการเข็นรถไปตามเส้นแบ่งแผ่นรองพื้นตั้งแต่หมายเลข 1 ไปจนถึงหมายเลข 2 โดยให้สายอากาศตรงกับเส้นแบ่งแผ่นรองพื้นในแนวดิ่ง และทำการเข็นซ้ำอีก 10 รอบ เมื่อทำการทดสอบเสร็จสิ้นจะนำข้อมูลที่ได้มาประมวลผลเพื่อหาค่าความผิดพลาดของพิกัดตำแหน่ง โดยใช้โปรแกรม MATLAB ดังแสดงในรูปที่ 4.31 ถึงรูปที่ 4.33



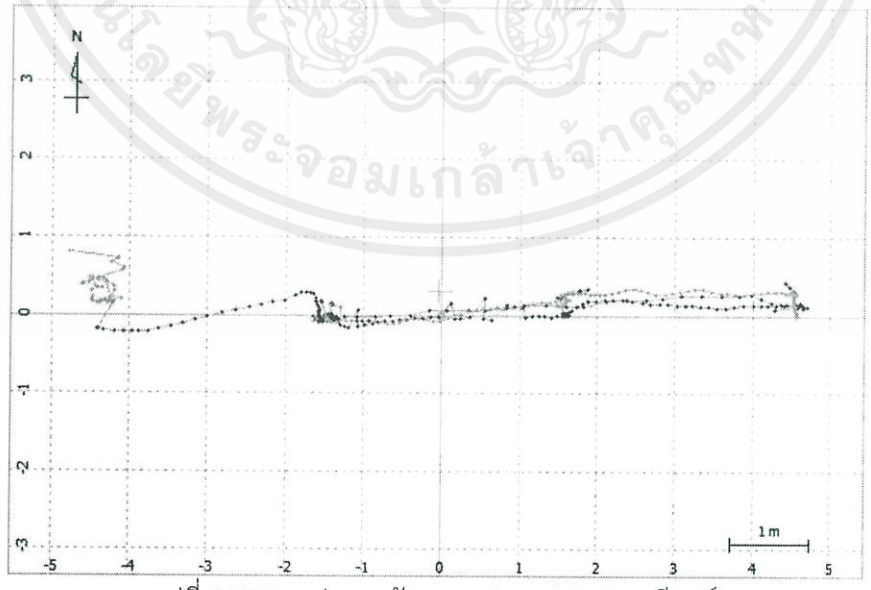
รูปที่ 4.31 กราฟแสดงพิกัดของสถานีจลน์เปรียบเทียบกับเส้นทางการเคลื่อนที่

จากรูปที่ 4.31 แสดงตำแหน่งของสถานีจลน์เปรียบเทียบกับพิกัดเส้นทางการเคลื่อนที่ โดยจุดสีแดงแสดงตำแหน่งของสถานีจลน์ เส้นสีม่วงแสดงเส้นทางการเดินรถเส้นที่ 1 เส้นสีเขียวแสดงเส้นทางการเดินรถเส้นที่ 2



รูปที่ 4.32 กราฟแสดงค่าความผิดพลาดของพิกัดตำแหน่ง

จากรูปที่ 4.32 แสดงค่าความผิดพลาดของพิกัดตำแหน่ง โดยแกน X แสดงค่าเฉลี่ยของพิกัดตำแหน่งในการเดินรถ แกน Y แสดงค่าความผิดพลาดของพิกัดตำแหน่งในหน่วยเมตร โดยพิจารณาค่าความผิดพลาดที่ตำแหน่ง 95 เปอร์เซ็นไทล์ จะพบว่าได้ค่าความผิดพลาดอยู่ที่ 0.022 เมตร

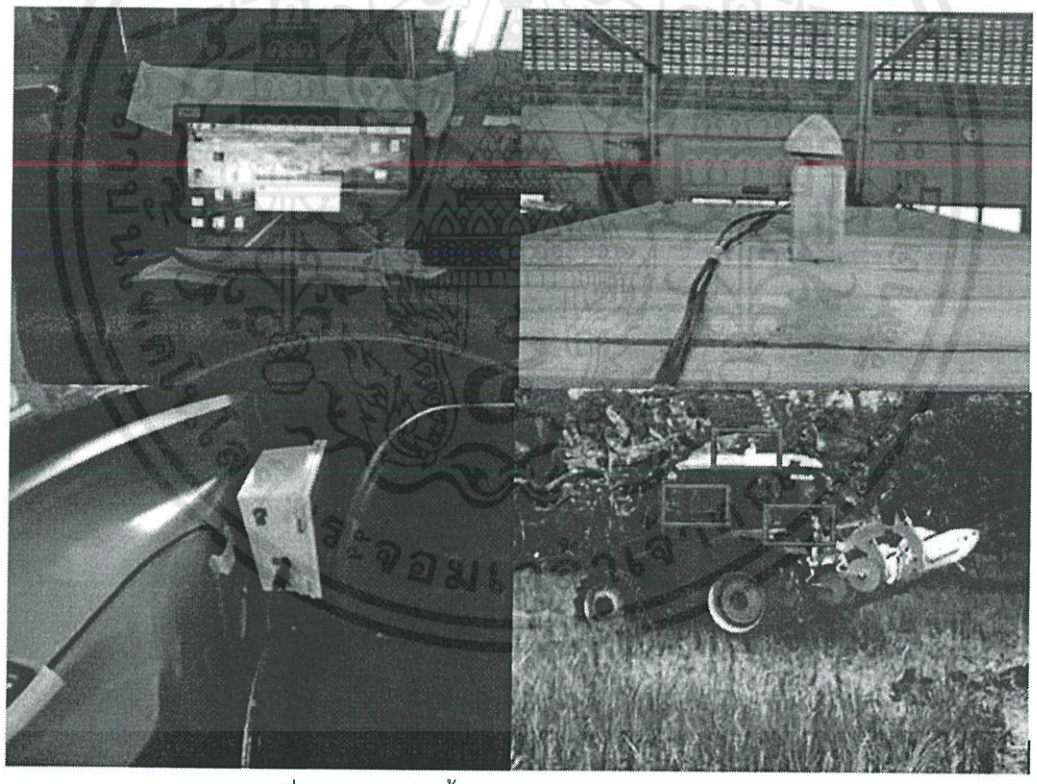


รูปที่ 4.33 กราฟแสดงอัตราการ Fixed ของสถานีจลน์

จากรูปที่ 4.33 แสดงอัตราการ Fixed ของสถานีจลน์ โดยการ Fixed ของสถานีจลน์หมายถึงสถานีจลน์สามารถรับข้อมูลพิกัดตำแหน่งจากสถานีฐานได้ ส่งผลให้มีความแม่นยำในการระบุตำแหน่งสูงในระดับเซนติเมตร ตามหลักการของเทคนิค RTK โดยกราฟจะแสดงสถานะของอุปกรณ์เมื่ออุปกรณ์อยู่ในสถานะ Fixed ด้วยจุดสีเขียว และแสดงสถานะของอุปกรณ์ เมื่ออุปกรณ์ไม่อยู่ในสถานะ Fixed ด้วยจุดสีส้ม จากนั้นนำสถานะที่ได้มาคำนวณอัตราการ Fixed ซึ่งมีค่าอัตราการ Fixed ร้อยละ 89.3

4.5.1 ผลการทดสอบในแปลงเกษตร

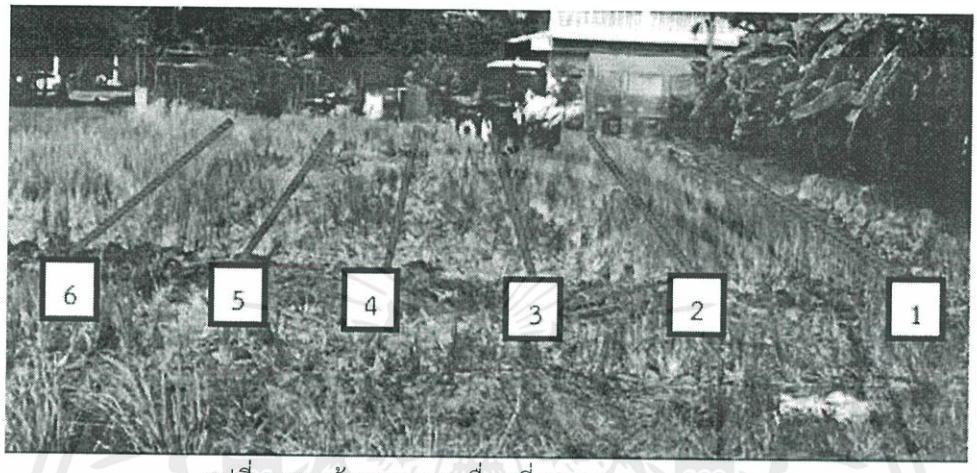
การทดสอบในแปลงเกษตรเริ่มจากการติดตั้งอุปกรณ์บนรถทางเกษตรกรรมยี่ห้อ NEWHOLLAND ดังแสดงในรูปที่ 4.34



รูปที่ 4.34 การติดตั้งสถานีจลน์บนรถทางการเกษตร

จากรูปที่ 4.34 การติดตั้งสถานีจลน์บนรถทางเกษตรกรรมทำได้โดยการติดตั้งสายอากาศที่บริเวณหลังคารถ ติดตั้งหน้าจอแสดงผลบนคอนโซลหน้ารถ และติดตั้งส่วนประมวลผลบริเวณ

ด้านหลังคนขับ เมื่อทำการติดตั้งเสร็จสิ้นแล้ว ทำการสร้างเส้นทางการเคลื่อนที่บนแปลงเกษตรดังแสดงในรูปที่ 4.35

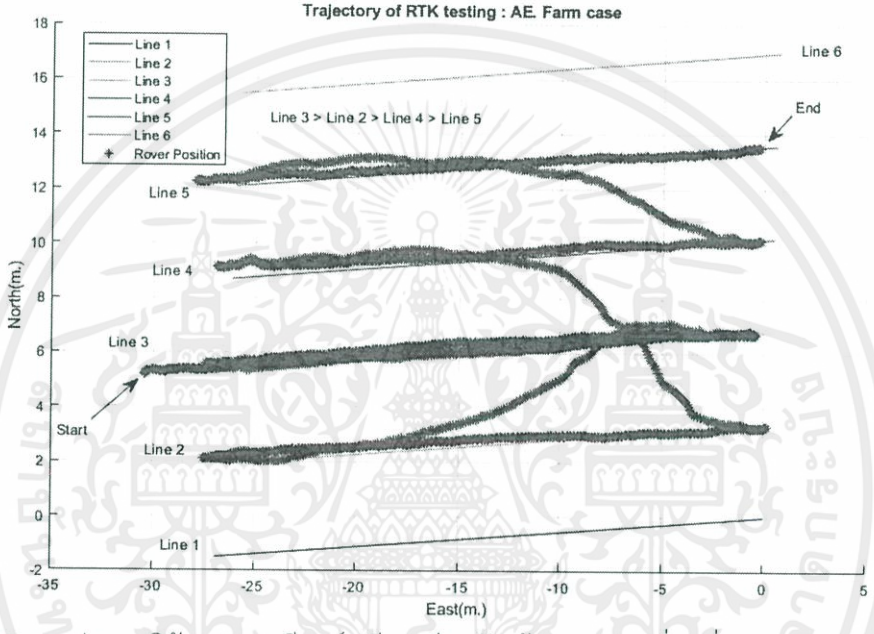


รูปที่ 4.35 เส้นทางการเคลื่อนที่ของรถทางเกษตรกรรม

จากรูปที่ 4.35 แสดงเส้นทางการเคลื่อนที่ของรถทางเกษตรกรรม โดยมีเส้นทางจำนวนทั้งสิ้น 6 เส้นทาง จากนั้นทำการปักเสาบริเวณจุดต้นทางและปลายทางของเส้นทางทุกเส้น เพื่อใช้เป็นจุดอ้างอิงสำหรับการเคลื่อนที่ของรถทางเกษตรกรรม แล้วทำการเดินรถทางเกษตรกรรม เพื่อทำการทดสอบประสิทธิภาพการทำงานของสถานีจลน์ ดังแสดงในรูปที่ 4.36

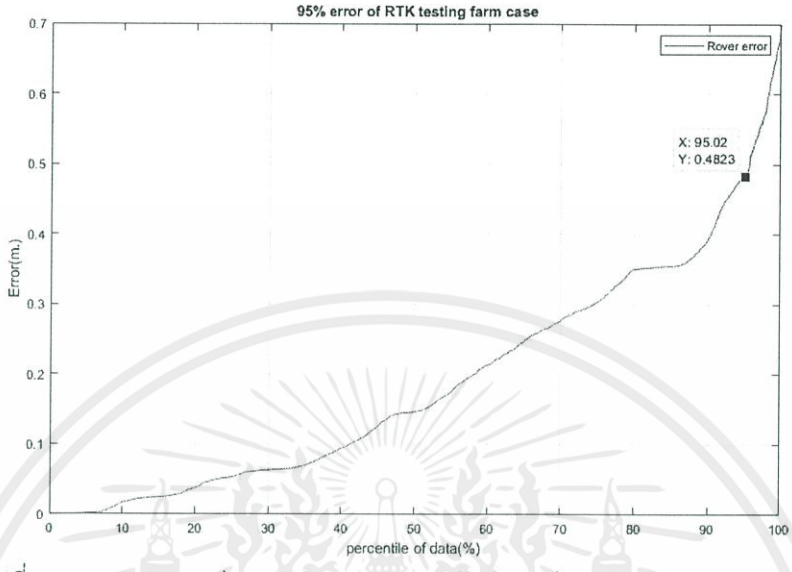
รูปที่ 4.36 การเดินรถทางเกษตรกรรม

จากรูปที่ 4.36 แสดงการเคลื่อนที่ของรถทางเกษตรกรรม ซึ่งเคลื่อนที่ตามแนวเสา และไถหน้าดินเป็นเส้นตรงจำนวน 4 เส้นทาง โดยไถหน้าดินเส้นทางที่ 2 ถึงเส้นทางที่ 5 เนื่องจากเส้นทางที่ 1 และ 6 เป็นแนวขอบของไร่ จากนั้นนำค่าพิกัดที่ได้จากกการทดสอบได้มาประมวลผลเพื่อหาค่าความผิดพลาดของพิกัดตำแหน่ง โดยใช้โปรแกรม MATLAB ดังแสดงในรูปที่ 4.37 ถึงรูปที่ 4.39



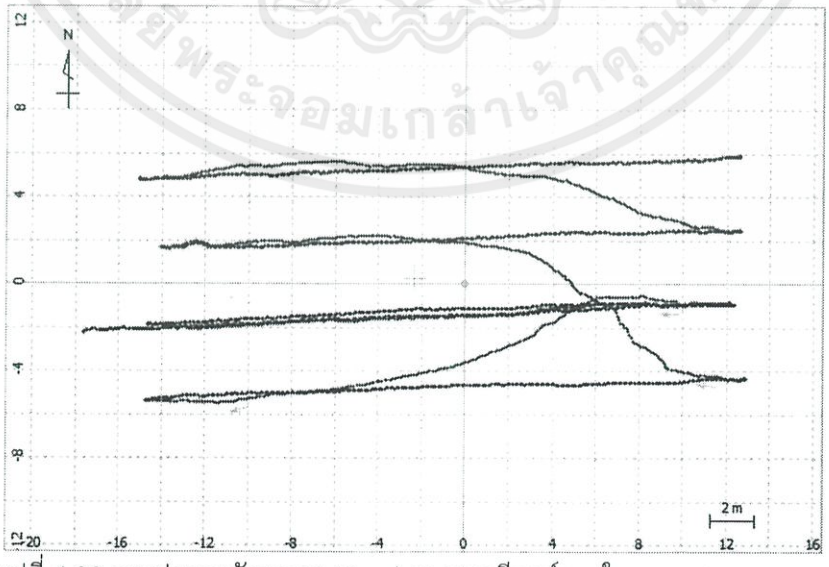
รูปที่ 4.37 กราฟแสดงพิกัดของสถานีจลน์เปรียบเทียบกับเส้นทางการเคลื่อนที่ของรถทางเกษตรกรรม

จากรูปที่ 4.37 แสดงตำแหน่งของสถานีจลน์เปรียบเทียบกับพิกัดเส้นทางการเคลื่อนที่ โดยจุดสีแดงแสดงตำแหน่งของสถานีจลน์ และเส้นทางการเคลื่อนที่แสดงด้วยเส้นตรงสีต่างๆ รวมทั้งสิ้น 6 เส้นทาง



รูปที่ 4.38 กราฟแสดงค่าความผิดพลาดของพิกัดตำแหน่งของรถทางเกษตรกรรม

จากรูปที่ 4.38 แสดงค่าความผิดพลาดของพิกัดตำแหน่งของรถทางเกษตรกรรม โดยแกน X แสดงค่าเฉลี่ยของพิกัดตำแหน่งในการเดินรถ แกน Y แสดงค่าความผิดพลาดของพิกัดตำแหน่งในหน่วยเมตร โดยพิจารณาค่าความผิดพลาดที่ตำแหน่ง 95 เปอร์เซ็นไทล์ จะพบว่าได้ค่าความผิดพลาดอยู่ที่ 0.4823 เมตร โดยการพิจารณาการแนวเดินรถทางเกษตรกรรม พิจารณาจากแนวคันไถทางด้านหลังรถ ซึ่งมีระยะเอียงออกไปจากตัวรถ 40 เซนติเมตรส่งผลให้เกิดการเลื่อนของพิกัดตำแหน่ง 40 เซนติเมตร ดังนั้นค่าความผิดพลาดจริงจึงอยู่ที่ 0.4823 – 0.40 เมตร ซึ่งก็คือ 0.0823 เมตร



รูปที่ 4.38 กราฟแสดงอัตราการ Fixed ของสถานีจลน์ภายในรถทางเกษตรกรรม

จากรูปที่ 4.38 แสดงอัตราการ Fixed ของสถานีจลน์ โดยการ Fixed ของสถานีจลน์หมายถึงสถานีจลน์สามารถรับข้อมูลพิกัดตำแหน่งจากสถานีฐานได้ ส่งผลให้มีความแม่นยำในการระบุตำแหน่งสูงในระดับเซนติเมตร ตามหลักการของเทคนิค RTK โดยกราฟจะแสดงสถานะของอุปกรณ์เมื่ออุปกรณ์อยู่ในสถานะ Fixed ด้วยจุดสีเขียว และแสดงสถานะของอุปกรณ์ เมื่ออุปกรณ์ไม่อยู่ในสถานะ Fixed ด้วยจุดสีส้ม จากนั้นนำสถานะที่ได้มาคำนวณอัตราการ Fixed ซึ่งมีค่าอัตราการ Fixed ร้อยละ 99.4

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

จากการทำโครงงานฉบับนี้ทำการออกแบบสถานีจลน์ รวมถึงระบบเฝ้าสังเกตการณ์ทางหน้าจอเพื่อใช้ในการระบุพิกัดตำแหน่ง ประกอบด้วยส่วนการปรับแก้ค่าเซ็นเซอร์ การเลือกสถานีฐาน การสร้างเส้นทางทั้งแบบเก็บทีละจุด แบบเก็บที่ขอบไร่ทางเกษตรกรรม โดยแสดงเส้นทางผ่านแผนภาพ Canvas แบบเวลาจริง พร้อมทั้งระบุทิศทางที่ควรจะเคลื่อนที่ไปโดยสามารถแบ่งการสรุปออกได้ดังต่อไปนี้

5.1.1 ส่วนการออกแบบอุปกรณ์สถานีจลน์

การออกแบบอุปกรณ์สถานีจลน์ เป็นการออกแบบการจัดวางอุปกรณ์ที่ใช้งาน ลงในบรรจุภัณฑ์ที่มีขนาดเล็ก โดยใช้แหล่งจ่ายไฟเป็น Power Bank ขนาด 5 โวลต์ 20 แอมป์ชั่วโมง และมีการต่อจอแสดงผล LCD แบบสัมผัสขนาด 7 นิ้ว เพื่อความสะดวกในการใช้งาน นอกจากนี้ยังมีการออกแบบอุปกรณ์ติดตั้งสายอากาศที่มีขนาดเล็ก และสะดวกต่อการติดตั้งเข้ากับกับรถทางเกษตรกรรมแบบแทนที่เสาสัญญาณที่มีขนาดใหญ่ ส่งผลให้อุปกรณ์พกพาสะดวก ใช้งานง่าย และมีประสิทธิภาพ

ในส่วนต่อประสานกราฟิกกับผู้ใช้เป็นการออกแบบการใช้งานอุปกรณ์ Raspberry Pi ร่วมกับจอแสดงผล LCD แบบสัมผัสขนาด 7 นิ้ว โดยมี Icon สำหรับเรียกใช้โปรแกรม RTKRCV สำหรับคำนวณตำแหน่งด้วยเทคนิค RTK และโปรแกรมภาษา Python สำหรับเฝ้าสังเกตการณ์ตำแหน่งของผู้ใช้งานทางหน้าจอ และสามารถเรียกใช้โปรแกรมโดยง่าย เพียงแค่สัมผัสรูป Icon บนหน้าจอ

จากผลการทดสอบการทำงานของสถานีจลน์ พบว่าสามารถสร้างเส้นทางการเดินรถได้อย่างถูกต้อง โดยมีรูปแบบการสร้างเส้นทางการเดินรถสองแบบ ได้แก่ เส้นทางแบบเก็บค่าพิกัดการเดินรถแต่ละจุด และแบบกำหนดพิกัดอ้างอิง 4 จุด แล้วสร้างเส้นทางการเดินรถระหว่างพิกัดอ้างอิงที่กำหนด หลังจากสร้างเส้นทางเสร็จแล้ว จึงทดสอบการทำงานในส่วนของการระบุพิกัดขณะเดินรถทางเกษตรกรรม โดยมีความแม่นยำของการระบุตำแหน่งไม่เกิน 5 เซนติเมตร เมื่อเปรียบเทียบกับพิกัดอ้างอิงจากสถานีฐาน

จากผลการทดสอบการเคลื่อนที่ของรถทางเกษตรกรรมแบบอัตโนมัติ พบว่าสามารถขับเคลื่อนได้แบบเรียลไทม์ และใช้งานได้จริงตามรูปแบบที่กำหนดไว้ แต่มีข้อผิดพลาดด้านการควบคุมการเลี้ยวได้ เนื่องจากรูปแบบการเลี้ยวเป็นรูปแบบเดียวกับรถล้อตะขาบ กล่าวคือ ใช้การเลี้ยวแบบล้อ

ทั้งสองข้างหมุนด้วยความเร็วที่ต่างกัน ส่งผลให้การเลี้ยวไม่สามารถควบคุมการเลี้ยวได้ละเอียดเพียงพอ

5.1.2 การสร้างส่วนต่อประสานกราฟิกกับผู้ใช้

ในส่วนต่อประสานกราฟิกกับผู้ใช้ เขียนด้วยภาษา Python เพื่อใช้งานบน Raspberry Pi แบ่งออกเป็น 3 ส่วนหลัก

1) ส่วนการตั้งค่าโปรแกรม เมื่อทำการเลือกสถานีฐาน ที่ต้องการตั้งค่าลงในส่วนต่อประสานกราฟิกกับผู้ใช้ โปรแกรมที่เขียนขึ้นสามารถนำค่าอินพุตดังกล่าวไปแก้ไข Configuration file ได้อย่างถูกต้อง และนำไปใช้ประมวลผลการรับค่าพิกัดตำแหน่งด้วยโปรแกรม RTKRCV ได้ โดยสามารถเลือกสถานีฐานได้โดยการกดปุ่ม และมีการตั้งค่าปรับแก้ของเซ็นเซอร์ โดยป้อนความสูงของสายอากาศในหน่วยเมตร และทำการขยับสายอากาศเป็นเวลา 30 วินาที เพื่อให้เซ็นเซอร์พร้อมในการแพ้ค่าพิกัดความผิดพลาดอันเกิดจากการเอียงตัวของสายอากาศ

2) ส่วนแสดงเส้นทางเดินรถทางเกษตรกรรม เมื่อกดปุ่มเพื่อเก็บค่าพิกัดทั้งในโหมด Single Waypoint คือการกดทีละจุด และแบบ Multiple Waypoint คือการเก็บจุดพิกัดที่มุมของไร่ทางเกษตรกรรม จะสามารถแสดงเส้นทางการเดินรถทางเกษตรกรรมบนหน้า Canvas และแสดงเส้นทางนี้บน Google Maps ได้

3) ส่วนแสดงแผนที่และการนำทาง จะทำการแสดงผลพิกัดตำแหน่งที่รับค่ามาจากสายอากาศแบบเวลาจริง มาปรับแก้ค่าความเอียงโดยใช้ค่า Roll Pitch และYaw จากเซ็นเซอร์ โดยสามารถเลือกได้ว่าจะใช้เซ็นเซอร์ปรับแก้ค่าความเอียงหรือไม่ หากมีการเปิด-ปิด ของเซ็นเซอร์ระหว่างการทำงาน โปรแกรมจะยังคงแสดงผลต่อไปได้อย่างต่อเนื่อง และมีการแสดงระยะห่างระหว่างรถทางเกษตรกรรมกับเส้นทาง เวลา และทิศทาง เพื่อให้ช่วยในการนำทางอย่างมีประสิทธิภาพ

5.2 ข้อเสนอแนะ

1) การสร้างจุดปักหมุดสำหรับการคำนวณตำแหน่งด้วยเทคนิค RTK ควรปักหมุดด้วยหลักการสำรวจ ทำแผนที่ และร่างเส้นเชื่อมระหว่างจุดปักหมุดโดยใช้เครื่องดีเส้น

2) ควรพัฒนาส่วนต่อประสานกราฟิกกับผู้ใช้ให้สามารถเชื่อมต่อกับสถานีฐานของกรมที่ดินได้ โดยไม่ต้องพิมพ์คำสั่งเพิ่ม

3) ในส่วนต่อประสานกราฟิกกับผู้ใช้ ส่วนของการแสดงแผนที่และการนำทาง ควรที่จะสามารถขยายรูปของเส้นทางบน Canvas ได้

4) พัฒนาการติดตั้งสายเชื่อมต่อให้สามารถใช้งานได้ง่ายยิ่งขึ้น และเพิ่มเติมส่วนการแสดงปริมาณคงเหลือของแบตเตอรี่ ให้ผู้ใช้งานทราบ

5) ปรับปรุงการทำงานของรถทางเกษตรกรรมจำลอง ให้สามารถทำงานได้ราบรื่นยิ่งขึ้น

บรรณานุกรม

[1] Wikipedia. “ระบบกำหนดตำแหน่งบนโลก”
https://th.wikipedia.org/wiki/ระบบกำหนดตำแหน่งบนโลก
[2] Wikipedia. “ระบบกำหนดตำแหน่งบนโลก”
https://th.wikipedia.org/wiki/ระบบกำหนดตำแหน่งบนโลก
[3] Wikipedia. “ระบบกำหนดตำแหน่งบนโลก”
https://th.wikipedia.org/wiki/ระบบกำหนดตำแหน่งบนโลก
[4] Wikipedia. “ระบบกำหนดตำแหน่งบนโลก”
https://th.wikipedia.org/wiki/ระบบกำหนดตำแหน่งบนโลก
[5] Wikipedia. “ระบบกำหนดตำแหน่งบนโลก”
https://th.wikipedia.org/wiki/ระบบกำหนดตำแหน่งบนโลก
[6] Wikipedia. “ระบบกำหนดตำแหน่งบนโลก”
https://th.wikipedia.org/wiki/ระบบกำหนดตำแหน่งบนโลก
[7] Wikipedia. “พิกัดภูมิศาสตร์”
https://www.prosoftgps.com/Article/Detail/72143
[8] Wikipedia. “Haversine formula”
https://www.igismap.com/haversine-formula-calculate-geographic-distance-earth/
[9] Wikipedia. “ระบบพิกัดกริดแบบยูทีเอ็ม”
https://gisguru.wordpress.com/2009/11/09/ระบบพิกัดแผนที่/
[10] Gistda. “ระบบพิกัดในแผนที่”
https://www.gistda.or.th/main/th/node/873
[11] cbalch. “Relationship between Kp and the Aurora”
http://www.spaceweather.com/glossary/kp.html
[12] หลักการมอเตอร์ไฟฟ้ากระแสตรง
http://tisade.blogspot.com/
[13] หลักการแบตเตอรี่
https://www.klangbattery.com/how_to.html
[14] หลักการไอซี L298N

https://www.arduinoall.com/product/571/l298n-ไอซีขับมอเตอร์-l298n-full-bridge-dual-motor-driver-l298-ic

[15] ระบบฐานข้อมูล

http://www.glurgeek.com/education/ระบบฐานข้อมูล-database-system-คือ-ระบบ/

[16] HTML

http://www.codingbasic.com/html.html

[17] หลักการสร้างส่วนต่อประสานกราฟิกกับผู้ใช้

https://th.wikipedia.org/wiki/ส่วนต่อประสานกราฟิกกับผู้ใช้

[18] เครื่องรับสัญญาณจีเอ็นเอสเอสแบบความถี่เดียวยี่ห้อ U-blox NEO M8T

https://www.u-blox.com/en/product/neolea-m8t-series

[19] สายอากาศ TW-3740

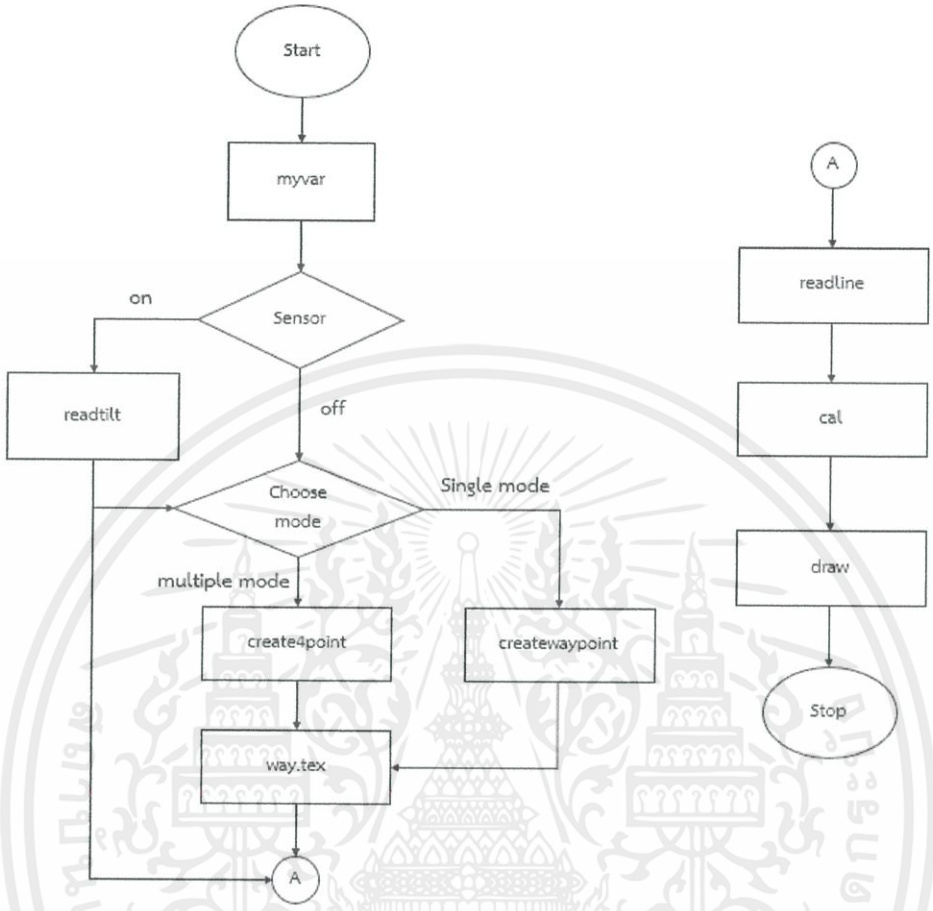
http://www.tallysman.com/index.php/gnss/products/antennas-gpsbeidougalileoglonass/tw3740-tw3742/

[20] ภาษาไพธอน

https://saixiii.com/python-programming/

ภาคผนวก ก

แผนภาพการทำงานของโปรแกรม



รูป ภาคผนวก ก ที่ 1 แผนภาพแสดงการทำงานรวมของโปรแกรม

จากรูป ภาคผนวก ก ที่ 1 แสดงแผนภาพการทำงานของโปรแกรม เริ่มจากการรับค่าพิกัดตำแหน่งของ rover โดยใช้ฟังกชั่น myvar จากนั้นจะทำการตรวจสอบว่ามีการใช้งาน sensor หรือไม่ หากมีการใช้งานเซนเซอร์ จะใช้งานฟังกชั่น readtilt ซึ่งเป็นฟังก์ชันปรับเทียบค่าพิกัดตำแหน่งโดยอ้างอิงจากค่าพิกัดความเอียงสายอากาศ หากไม่มีการใช้งาน sensor โปรแกรมจะนำค่าพิกัดตำแหน่งที่ไม่ได้ปรับเทียบไปยังฟังก์ชัน Choose mode โดยจะมีโหมดการทำงาน 2 โหมด ได้แก่ single mode และ multiple mode หลังจากเลือกโหมดการทำงานแล้ว โปรแกรมจะกำหนดเส้นทางการเดินรถทางเกษตรกรรมผ่านฟังก์ชัน createwaypoint และ create4point ตามลำดับ เมื่อกำหนดเส้นทางได้แล้ว ค่าพิกัดเส้นทางจะถูกเก็บไว้ในไฟล์ way.txt จากนั้นใช้ฟังกชั่น readline ในการเลือกแนวการเดินรถทางเกษตรกรรม โดยพิจารณาแนวการเดินรถจากค่าพิกัดตำแหน่งของ rover เปรียบเทียบกับพิกัดเส้นทางในไฟล์ way.txt เมื่อพิจารณาเรียบร้อยแล้ว จะใช้งานฟังก์ชัน cal เพื่อหาระยะห่างระหว่าง rover และเส้นทาง จากนั้นใช้ฟังก์ชัน draw ในการวาดเส้นทางทั้งหมดบนส่วนต่อประสานกราฟิกกับผู้ใช้

ภาคผนวก ข

โปรแกรมที่ใช้อ่านค่ามุมแสดงการเคลื่อนที่ของวัตถุโดยใช้เซนเซอร์ GY-85

โปรแกรมที่ใช้อ่านค่ามุมของเซนเซอร์ รุ่น GY-85

```
import time
import struct
import sys
import math
import pigpio
savecompass(cx,cy,cz)
file = open("Compass.txt","a")
data =str([cx,cy,cz])
file.write(data+'\n')
file.write('%.4f,%.4f,%.4f\n' % (cx,cy,cz))
return
if sys.version > '3':
buffer = memoryview
BUS=1
ADXL345_I2C_ADDR=0x53
HMC5883L_I2C_ADDR=0x1E
RUNTIME=60.
pi=pigpio.pi()
h = pi.i2c_open(BUS, ADXL345_I2C_ADDR)
c = pi.i2c_open(BUS, HMC5883L_I2C_ADDR)
if (h >= 0) and (c >= 0)
# Initialise ADXL345.
pi.i2c_write_byte_data(h, 0x2d, 0)  # POWER_CTL reset.
pi.i2c_write_byte_data(h, 0x2d, 8)  # POWER_CTL measure.
pi.i2c_write_byte_data(h, 0x31, 0)  # DATA_FORMAT reset.
pi.i2c_write_byte_data(h, 0x31, 11) # DATA_FORMAT full res +/- 16g.
# Initialise HMC5883L.
pi.i2c_write_byte_data(c, 0x00, 0xF0)  # DATA_FORMAT reset.
pi.i2c_write_byte_data(c, 0x02, 0x00) # DATA_FORMAT full res +/- 16g.
read = 0
start_time = time.time()
while 1:
(s, b) = pi.i2c_read_i2c_block_data(h, 0x32, 6)
(T, V) = pi.i2c_read_i2c_block_data(c, 0x03, 6)

if (s >= 0) and (T >= 0):
(x, y, z) = struct.unpack('<3h', buffer(b))
(cx ,cy, cz) = struct.unpack('>3h', buffer(V))
savecompass(cx,cy,cz)
cx = cx-139
cz = cz+196
roll = math.atan2(y, math.sqrt(x * x + z * z)) * 180 / math.pi
pitch =-1* math.atan2(x, math.sqrt(y * y + z * z)) * 180 / math.pi
yaw = math.atan2(cz,cx) * 180 / math.pi
if yaw < 0:
yaw = yaw+360
#print("{} {} {}".format(cx, cy, cz))
print( '%.2f' %roll, '%.2f' %pitch , '%.2f' %(yaw))
read += 1
#pi.i2c_close(h)
#pi.i2c_close(c)
#pi.stop()
#print(read, read/RUNTIME)
```

ภาคผนวก ค

ชุดคำสั่งส่วนต่อประสานกราฟิกกับผู้ใช้ ด้วยภาษาไพธอน

ชุดคำสั่งส่วนต่อประสานกราฟิกกับผู้ใช้ ประกอบไปด้วย ส่วนการกำหนดค่า ส่วนการแสดงพิกัดตำแหน่งบนแผนที่ และส่วนแจ้งเตือนเมื่อออกนอกเส้นทาง

```
import subprocess, gpxpy.geo, math, string, socket, time, os, utm
from tkinter import *
import pigpio, struct
import tkinter as tk
from tkinter import ttk
import parser
import time
from PIL import Image, ImageDraw
import numpy as np
import RPi.GPIO as gpio
import sys
# import class and funtion
turnbacklr = 0
gpio.setmode(gpio.BOARD)
gpio.setup(11, gpio.OUT)
gpio.setup(13, gpio.OUT)
gpio.setup(15, gpio.OUT)
gpio.setup(19, gpio.OUT)
gpio.setup(29, gpio.OUT)
gpio.setup(31, gpio.OUT)
mypwm1 = gpio.PWM(29, 100)
mypwm2 = gpio.PWM(31, 100)
mypwm1.start(0)
mypwm2.start(0)

page = 1
width = 300
height = 300
# Kalmath.n Paramath.ter
```

```
dt = 0.02
predicpitch = 0
predicroll = 0
p00 = 0.1
p01 = 0.1
p11 = 0.1
R = 5
Q = 0.1
#global axi,ayi,azi,gxi,gyi,gzi,cxsf,cysf,cxof,cyof
running = False  # Global flag
################  sensor zone ######################
BUS=1
pi=pigpio.pi() # open local Pi
# pin address to sensor
ADXL345_I2C_ADDR=0x53
HMC5883L_I2C_ADDR=0x1E
ITG3205_I2C_ADDR=0x68

try : # check sensor on or off
    #global h,c,g
    h = pi.i2c_open(BUS, ADXL345_I2C_ADDR)
    c = pi.i2c_open(BUS, HMC5883L_I2C_ADDR)
    g = pi.i2c_open(BUS, ITG3205_I2C_ADDR)
    #print(h,c,g)
    global s
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    if sys.version > '3':
        buffer = memoryview
    if (h >= 0) and (c >= 0) and (g>=0):
   # Initialise ADXL345.
        pi.i2c_write_byte_data(h, 0x2d, 0)  # POWER_CTL reset.
```

```
        pi.i2c_write_byte_data(h, 0x2d, 8)  # POWER_CTL math.asure.
        pi.i2c_write_byte_data(h, 0x31, 0)  # DATA_FORMAT reset.
        pi.i2c_write_byte_data(h, 0x31, 11) # DATA_FORMAT full res +/- 16g.
    # Initialise HMC5883L.
        pi.i2c_write_byte_data(c, 0x00, 0xF0)  #
        pi.i2c_write_byte_data(c, 0x02, 0x00) #
    # Initialise ITG3205
        pi.i2c_write_byte_data(g, 0x3E, 0x01)
        pi.i2c_write_byte_data(g, 0x15, 0x07)
        pi.i2c_write_byte_data(g, 0x16, 0x18| 0x01)
        pi.i2c_write_byte_data(g, 0x17, 0x00)
        read = 0
except pigpio.error:
    print('Sensor Failed')
    pass#continue


################################# IMU Sensor Connected OK?
#######################################


def savecalibrate(axi, ayi, azi, gxi, gyi, gzi, cxmax, cxmin, cymax, cymin, cxsf, cysf, cxof,
cyof):
    file = open("Calib_log.txt", "a")
    date = str(time.asctime(time.localtime(time.time())))
    file.write(
        '%s | Axi Ayi Azi =  %i %i %i, Gxi Gyi Gzi = %i %i %i, cxmax cxmin = %i %i, cymax
cymin = %i %i, cxsf = %.2f, cysf = %.2f, cxof = %.2f, cyof = %.2f\n' % (
        date, axi, ayi, azi, gxi, gyi, gzi, cxmax, cxmin, cymax, cymin, cxsf, cysf, cxof, cyof))
    return
# save calibrate sensor data
```

```
def savecompass(cx, cy, cz):
    file = open("Compass.txt", "a")
    file.write('%.4f,%.4f,%.4f\n' % (cx, cy, cz))
    return
# save compass of sensor

def savecalib(roll,pitch,yaw):
    file = open("Calib.txt","a")
    file.write('%.4f,%.4f,%.4f\n' % (roll,pitch,yaw))
    return
# save roll,pitch,yaw
def savesol(timer,lat,lng,h,ratio,disty,side):
#savesol(duration,lat,lng,ratio,distance,sidecar)
    file = open("Sol.pos","a")

    timersp = timer.split(" ")
    timer1=timersp[0]
    timer2=timersp[1]
    #print(timer1,timer2)
    file.write('%s,%s,%.8f,%.8f,%.3f,%.1f,%.3f,%s\n' %
(timer1,timer2,lat,lng,h,ratio,disty,side))
    return
# save solution of rtk technique
def saveaccel(ax, ay, az):
    file = open("Accel.txt", "a")
    file.write('%.4f,%.4f,%.4f\n' % (ax, ay, az))
    return
# save accel of sensor

def savegyro(gx, gy, gz):
```

```
    file = open("Gyro.txt", "a")
    file.write('%.4f,%.4f,%.4f\n' % (gx, gy, gz))
    return
# save gyro of sensor
def bufferdraw(lat,lng,q):
    file = open("drawbuffer.txt", "a+")
    file.truncate(0)
    file.write('%s %s %s' % (lat,lng,q))
    return
# save latitude longitude quality
def readbufferdraw():
    file = open("drawbuffer.txt", "r+")
    data = (file.read()).split(" ")
    lat = float(data[0])
    lng = float(data[1])
    q = int(data[2])
    return lat, lng, q
# read latitude longitude quality

def bufferauto(Distance,side):
    file = open("autobuffer.txt", "a+")
    file.truncate(0)
    file.write('%s %s' % (Distance,side))
    return
# save distance and side of rover
def readbufferauto():
    file = open("autobuffer.txt", "r+")
    data = (file.read()).split(" ")
    disty = data[0]
    side = data[1]
  # read distance and side of rover
```

```
def Logfile2(timer, lat, lng, h, q, sat, sde, sdn, sdu, sden, sdnu, sdeu, age, ratio):
    f = open('Sol_raw.txt', 'a+')
    f.write('%s  %.8f  %.8f\t%.4f  %i  %i  %.4f  %.4f  %.4f  %.4f  %.4f  %.4f  %.2f
%.1f\n' % (
    timer, lat, lng, h, q, sat, sde, sdn, sdu, sden, sdnu, sdeu, age, ratio))
    f.close()
# save timer, lat, lng, h, q, sat, sde, sdn, sdu, sden, sdnu, sdeu, age, ratio

def buffercalib(axi, ayi, azi, gxi, gyi, gzi, cxsf, cysf, cxof, cyof):
    fb = open('/home/pi/Desktop/program/buffercalib.txt', 'w+')
    fb.write('%i,%i,%i,%i,%i,%i,%.2f,%.2f,%.2f,%.2f' % (axi, ayi, azi, gxi, gyi, gzi, cxsf, cysf,
cxof, cyof))
    fb.close()
# save axi, ayi, azi, gxi, gyi, gzi, cxsf, cysf, cxof, cyof

def readbuffercalib():
    fb = open('/home/pi/Desktop/program/buffercalib.txt', 'r')
    data = (fb.read()).split(",")
    fb.close
    return data

def readsensor(h, c, g):
    h=int(h)
    c=int(c)
    g=int(g)
    (A, b) = pi.i2c_read_i2c_block_data(h, 0x32, 6)
    (C, V) = pi.i2c_read_i2c_block_data(c, 0x03, 6)
    (G, O) = pi.i2c_read_i2c_block_data(g, 0x1D, 6)
    (ax, ay, az) = struct.unpack('<3h', buffer(b)) #acc
    (gx ,gy, gz) = struct.unpack('>3h', buffer(O)) #gyro
```

```
    (cx ,cy, cz) = struct.unpack('>3h', buffer(V)) #compass
    return [ax, ay, az, gx, gy, gz, cx, cy, cz]

def calibimu(*args):
    print("Start Calibration!, Rotate your sensor in 30 secs.")
    ts = time.time()
    ax0 = 0
    ay0 = 0
    az0 = 0
    gx0 = 0
    gy0 = 0
    gz0 = 0
    cxmax = 0
    cxmin = 0
    cymax = 0
    cymin = 0
    cx = 0
    cy = 0
    cxof = 0
    cyof = 0
    cxsf = 0
    cysf = 0
    meancx = 0
    meancy = 0
    flag = 0
    for ti in range(0, 500):
        rawimu = readsensor(h, c, g)
        ax0 += rawimu[0]
        ay0 += rawimu[1]
        az0 += rawimu[2]
        gx0 += rawimu[3]
```

```
        gy0 += rawimu[4]
        gz0 += rawimu[5]
        cx = rawimu[6]
        cy = rawimu[7]
        if cx > cxmax:
            cxmax = cx
        elif cx < cxmin:
            cxmin = cx
        if cy > cymax:
            cymax = cy
        elif cy < cymin:
            cymin = cy
        time.sleep(0.06)
    te = time.time()
    td = int(te - ts)  # wait time = sample*time.sleep
    axi = int(ax0 / 500)
    ayi = int(ay0 / 500)
    azi = int((az0 / 500) - 256)
    gxi = int(gx0 / 500)
    gyi = int(gy0 / 500)
    gzi = int(gz0 / 500)
    meancx = (cxmax - cxmin) / 2
    meancy = (cymax - cymin) / 2
    cxsf = (cymax - cymin) / (cxmax - cxmin)
    if abs(cxsf) < 1:
        cxsf = 1
    cysf = (cxmax - cxmin) / (cymax - cymin)
    if abs(cysf) < 1:
        cysf = 1
    cxof = (meancx - cxmax) * cxsf
    cyof = (meancy - cymax) * cysf
```

```
    print("Time to calibrate = %i secs." % int(td))
    print("Calibration Done!")
    savecalibrate(axi, ayi, azi, gxi, gyi, gzi, cxmax, cxmin, cymax, cymin, cxsf, cysf, cxof,
cyof)
    buffercalib(axi, ayi, azi, gxi, gyi, gzi, cxsf, cysf, cxof, cyof)
    return
# calibrate sensor

def readtilt(*args):  #
(axi,ayi,gxi,gyi,cxsf,cysf,cxof,cyof,predicroll,predicpitch,p00,p01,p11,Q,R):
    global dt, predicpitch, predicroll,yaw, p00, p01, p11, R, Q
    [axi, ayi, azi, gxi, gyi, gzi, cxsf, cysf, cxof, cyof] = readbuffercalib()
    rawimu = readsensor(h, c, g)
    # save raw data
    saveaccel(rawimu[0], rawimu[1], rawimu[2])
    savegyro(rawimu[3], rawimu[4], rawimu[5])
    savecompass(rawimu[6], rawimu[7], rawimu[8])
    # measurement roll and pitch from accel
    Acroll = math.atan2((rawimu[1] - int(ayi)) / 256,
                        (rawimu[2] - int(azi)) / 256) * 180 / math.pi  # rotate around x axis
    Acpitch = math.atan2(-(rawimu[0] - int(axi)) / 256, (
                rawimu[2] - int(azi)) / 256) * 180 / math.pi  # rotate around y axis in real
adc of x is composite
    td2 = dt
    # initial predict roll and pitch from gyro
    predicroll = predicroll + ((rawimu[3] - int(gxi)) / 14.375) * td2  # Acroll
    predicpitch = predicpitch + ((rawimu[4] - int(gyi)) / 14.375) * td2  # Acpitch
    p00 += td2 * (2 * p01 + dt * p11)
    p01 += td2 * p11
    p00 += td2 * Q
    p11 += td2 * Q
```

```
    kk0 = p00 / (p00 + R)
    kk1 = p01 / (p01 + R)
    # Measurement update
    predicroll += (Acroll - predicroll) * kk0
    predicpitch += (Acpitch - predicpitch) * kk0
    p00 *= (1 - kk0)
    p01 *= (1 - kk1)
    p11 -= kk1 * p01
    yaw = math.atan2((float(cysf) * rawimu[7]) + float(cyof), (float(cxsf) * rawimu[6]) +
float(cxof)) * 180 / math.pi
    if yaw < 0:
        yaw = yaw + 360
    #time.sleep(td)
    return Acroll, Acpitch, yaw #Acroll,Acpitch,yaw#
# edit position by imu

################################################# IMU zone
######################################################################
###############################################

################################ Coordinate Covertion
###########################################
def lla2xyz(lat, lon, alt):
    a = 6378137
    e = 8.1819190842622e-2
    lat = math.radians(lat)
    lon = math.radians(lon)
    N = a / math.sqrt(1 - math.pow(e, 2) * math.pow(math.sin(lat), 2))
    x = (N + alt) * math.cos(lat) * math.cos(lon)
    y = (N + alt) * math.cos(lat) * math.sin(lon)
    z = ((1 - math.pow(e, 2)) * N + alt) * math.sin(lat)
```

```
    return x, y, z
# convert llh to xyz
def xyz2lla(x, y, z):
    x = float(x)
    y = float(y)
    z = float(z)
    a = 6378137;
    e = 8.1819190842622e-2;
    asq = math.pow(a, 2);
    esq = math.pow(e, 2);
    b = math.sqrt(asq * (1 - esq));
    bsq = math.pow(b, 2);
    ep = math.sqrt((asq - bsq) / bsq);
    p = math.sqrt(math.pow(x, 2) + math.pow(y, 2));
    th = math.atan2(a * z, b * p);
    lon = math.atan2(y, x);
    lat = math.atan2((z + math.pow(ep, 2) * b * math.pow(math.sin(th), 3)), (p - esq * a
* math.pow(math.cos(th), 3)));
    N = a / (math.sqrt(1 - esq * math.pow(math.sin(lat), 2)));
    alt = p / math.cos(lat) - N;
    lon = lon * 180 / math.pi;
    lat = lat * 180 / math.pi;
    return lat, lon, alt
# convert xyz to llh

def xyz2enu(latref, lngref, href, x, y, z):
    xr, yr, zr = lla2xyz(latref, lngref, href)
    lat = lat * math.pi / 180
    lng = lng * math.pi / 180
    slat = math.sin(lat)
    clat = math.cos(lat)
```

```
    slng = math.sin(lng)
    clng = math.cos(lng)
    e = -slng * (Xt - Xr) + clng * (Yt - Yr)
    n = -slat * clng * (Xt - Xr) - slat * slng * (Yt - Yr) + clat * (Zt - Zr)
    u = clat * clng * (Xt - Xr) + clat * slng * (Yt - Yr) + slat * (Zt - Zr)
    return e, n, u
# convert xyz to enu

def enu2xyz(latref, lngref, href, e, n, u):
    xr, yr, zr = lla2xyz(latref, lngref, href)
    lat = lat * math.pi / 180
    lng = lng * math.pi / 180
    slat = math.sin(lat)
    clat = math.cos(lat)
    slng = math.sin(lng)
    clng = math.cos(lng)
    x = -slng * e - clng * slat * n + clng * clat * u + xr
    y = clng * e - slng * slat * n + clat * slng * u + yr
    z = clat * n + slat * u + zr
    return x, y, z
# convert enu to xyz

def haversine(lat1, lon1, lat2, lon2):
    R = 6371000
    dLat = math.radians(lat2 - lat1)
    dLon = math.radians(lon2 - lon1)
    lat1 = math.radians(lat1)
    lat2 = math.radians(lat2)
    a = math.sin(dLat / 2) ** 2 + math.cos(lat1) * math.cos(lat2) * math.sin(dLon / 2) **
2
    c = 2 * math.asin(math.sqrt(a))
```

```
    return R * c
# find distance between point

def compensate(lat,lon,alt,r2, p2, y2, h):  # input r2,p2,y2
    # enu initial
    Ei = 0

    Ni = 0
    Ui = float(h)
    # Rot math.trix
    r = r2
    p = -p2
    y = 90 - y2
    cr = math.cos((r * math.pi) / 180)
    sr = math.cos((r * math.pi) / 180)
    cp = math.cos((p * math.pi) / 180)
    sp = math.cos((p * math.pi) / 180)
    cy = math.cos((y * math.pi) / 180)
    sy = math.cos((y * math.pi) / 180)
    # enu adjust
    Ead = cp * cy * Ei + (sr * sp * cy - cr * sy) * Ni + (cr * sp * cy + sr * sy) * Ui
    Nad = cp * sy * Ei + (sr * sp * sy - cr * cy) * Ni + (cr * sp * sy - sr * cy) * Ui
    Uad = -sp * Ei + sr * cp * Ni + cr * cp * Ui
    x,y,z=lla2xyz(lat, lon, alt)
    slat = math.sin((lat * math.pi) / 180)
    clat = math.cos((lat * math.pi) / 180)
    slng = math.sin((lon * math.pi) / 180)
    clng = math.cos((lon * math.pi) / 180)
    if abs(r2) > abs(p2):
        Ead = -Ead
        Nad = -Nad
```

```
        Uad = -Uad
    if abs(r2) < abs(p2):
        Ead = Ead
        Nad = Nad
        Uad = Uad
    Xc = -slng * Ead - clng * slat * Nad + clng * clat * Uad + x
    Yc = clng * Ead - slng * slat * Nad + clat * slng * Uad + y
    Zc = clat * Nad + slat * Uad + z
    latc,lngc,altc=xyz2lla(Xc, Yc, Zc)
    return latc,lngc,altc
# calculate position by sensor

################## Comath.ensation zone
#########################################################

def leavepos(lat, lng, latprelast, lngprelast):
    realpath = readline(lat, lng)
    # print(type(realpath))
    # print(type(realpath[0]))
    cal(lat, lng, float(realpath[0]), float(realpath[1]), float(realpath[2]), float(realpath[3]))
    distbetweenfloat = (gpxpy.geo.haversine_distance(lat, lng, latprelast, lngprelast))
    return distbetweenfloat
# find distance of point in 1 hz

############### Pomath.y zone
#################################################
def index(lat, lng, latstart, longstart, latfinal, longfinal):
    dist1 = gpxpy.geo.haversine_distance(latfinal, longfinal, latstart, longstart)
    dist2 = gpxpy.geo.haversine_distance(latfinal, longfinal, lat, lng)
    dist3 = gpxpy.geo.haversine_distance(latstart, longstart, lat, lng)
    #index = dist2 + dist3
```

```
    #index = (((dist2 ** 2) + (dist3 ** 2)) ** 0.5) / dist1
    # print(dist1)
    s = (dist1+dist2+dist3)/2
    index = math.sqrt(s*abs(s-dist1)*abs(s-dist2)*abs(s-dist3))
    return index
# find index of each waypoint

def readline(lat, lng):
    global realpath
    #dd = checkmode()
    f = open("way.txt", "r")
    file = f.readlines()
    L = round(len(file))
    path = []
    checksingle = open("/home/pi/Desktop/program/waysingle.txt")
    singlepath = len(checksingle.readlines())
    #print("SGP",singlepath)
    checkmuti = open("/home/pi/Desktop/program/way4point.txt")
    multipath = len(checkmuti.readlines())


    for j in range(1, L + 1):
        lat = float(lat)
        lng = float(lng)
        start = file[j - 1].split(' ')
        lats = float(start[0])
        lngs = float(start[1])

        if j < L:
            stop = file[j].split(' ')
            # print(stop)
```

```
        latsp = float(stop[0])
        lngsp = float(stop[1])
        path.append([lats, lngs, latsp, lngsp])

Lpath = len(path)
line = []
setindex = []
for k in range(1, Lpath):
    lats = path[k - 1][0]
    lngs = path[k - 1][1]
    latsp = path[k - 1][2]
    lngsp = path[k - 1][3]
    sline = index(lat, lng, lats, lngs, latsp, lngsp)
    setindex.append(sline)
    line.append([sline, lats, lngs, latsp, lngsp])
minline = min(setindex)
# realpath = [] # add tihs line
Lcheck = len(line)
for M in range(1, Lcheck + 1):
    if line[M - 1][0] == minline:
        realpath = [line[M - 1][1], line[M - 1][2], line[M - 1][3], line[M - 1][4]]
        dist2 = gpxpy.geo.haversine_distance(line[M - 1][1], line[M - 1][2], lat, lng)
        dist3 = gpxpy.geo.haversine_distance(line[M - 1][3], line[M - 1][4], lat, lng)

if singlepath == 2 :
    f = open("way.txt")
    data1 = f.readline()
    data11 = data1.split(' ')
    data2 = f.readline()
    data22 = data2.split(' ')
    f.close()
```

```
            cal(lat, lng, float(data11[0]), float(data11[1]), float(data22[2]), float(data22[3]))
            realpath=[float(data11[0]), float(data11[1]), float(data22[2]), float(data22[3])]
            print("realpath",realpath)
        else :
            cal(lat, lng, float(realpath[0]), float(realpath[1]), float(realpath[2]),
float(realpath[3]))
        return realpath
# find calculator path
def calforsave(lat, lng, latstart, longstart, latfinal, longfinal):
    # print(latstart, longstart, latfinal, longfinal)
    dist1 = gpxpy.geo.haversine_distance(latfinal, longfinal, latstart, longstart)
    dist2 = gpxpy.geo.haversine_distance(latfinal, longfinal, lat, lng)
    dist3 = gpxpy.geo.haversine_distance(latstart, longstart, lat, lng)
    disty = ((dist2 ** 2) - (dist1 - ((dist1 ** 2 - dist2 ** 2 + dist3 ** 2) / (2 * dist1))) ** 2) **
0.5
    disty = ('%.3f' % disty)
    utmstartref = utm.from_latlon(latstart, longstart)  # convert start point to utm
    utmfinalref = utm.from_latlon(latfinal, longfinal)  # convert end point to utm
    utmmeref = utm.from_latlon(lat, lng)  # convert tracking point to utm
    Xpath_start = utmstartref[0]
    Ypath_start = utmstartref[1]
    Xpath_end = utmfinalref[0]
    Ypath_end = utmfinalref[1]
    X_track = utmmeref[0]
    Y_track = utmmeref[1]
    dX_path = Xpath_end - Xpath_start
    dY_path = Ypath_end - Ypath_start
    dX_Track = X_track - Xpath_start
    dY_Track = Y_track - Ypath_start
    angle_path = math.degrees(math.atan2(float(dY_path), float(dX_path)))
    angle_track = math.degrees(math.atan2(float(dY_Track), float(dX_Track)))
```

```
    if angle_path < 0:
        angle_path = angle_path + 360
    if angle_track < 0:
        angle_track = angle_track + 360
    if (angle_track < angle_path) and (float(disty)>0.5):#1
        side = "turn_left"
    if (angle_track > angle_path) and (float(disty)>0.5):
        side = "turn_right"
    if (float(disty)< 0.1):
        side = "correct"
    return [disty,side]
# calculate distance between path and rover , side of approach to path
def cal(lat, lng, latstart, longstart, latfinal, longfinal):
    # print(latstart, longstart, latfinal, longfinal)
    dist1 = gpxpy.geo.haversine_distance(latfinal, longfinal, latstart, longstart)
    dist2 = gpxpy.geo.haversine_distance(latfinal, longfinal, lat, lng)
    dist3 = gpxpy.geo.haversine_distance(latstart, longstart, lat, lng)
    disty = ((dist2 ** 2) - (dist1 - ((dist1 ** 2 - dist2 ** 2 + dist3 ** 2) / (2 * dist1))) ** 2) **
0.5
    disty = ('%.3f' % disty)
    x7.set(disty)
    utmstartref = utm.from_latlon(latstart, longstart)  # convert start point to utm
    utmfinalref = utm.from_latlon(latfinal, longfinal)  # convert end point to utm
    utmmeref = utm.from_latlon(lat, lng)  # convert tracking point to utm
    Xpath_start = utmstartref[0]
    Ypath_start = utmstartref[1]
    Xpath_end = utmfinalref[0]
    Ypath_end = utmfinalref[1]
    X_track = utmmeref[0]
    Y_track = utmmeref[1]
    dX_path = Xpath_end - Xpath_start
```

```
    dY_path = Ypath_end - Ypath_start
    dX_Track = X_track - Xpath_start
    dY_Track = Y_track - Ypath_start
    angle_path = math.degrees(math.atan2(float(dY_path), float(dX_path)))
    angle_track = math.degrees(math.atan2(float(dY_Track), float(dX_Track)))

    if angle_path < 0:
        angle_path = angle_path + 360
    if angle_track < 0:
        angle_track = angle_track + 360

    disty2 = float(disty)

    if (angle_track < angle_path) :#       and (disty2 > 1)                    1
        canvasrealtime.delete("all")
        canvasrealtime.create_line(150-(15*disty2), 0, 150-(15*disty2), 300, fill="peru",
width=50)
        # canvas.create_line(250,50,220,20,220,80,250,50) #triangle right

canvasrealtime.create_polygon([140,240],[160,240],[163,255],[137,255],[140,240],fill="re
d")

canvasrealtime.create_polygon([135,255],[165,255],[165,295],[135,295],[135,255],fill="bl
ue")

canvasrealtime.create_polygon([130,255],[135,255],[135,265],[130,265],[130,255],fill="w
hite")

canvasrealtime.create_polygon([130,285],[135,285],[135,295],[130,295],[130,285],fill="w
hite")
```

```
canvasrealtime.create_polygon([165,255],[170,255],[170,265],[165,265],[165,255],fill="white")

canvasrealtime.create_polygon([165,285],[170,285],[170,295],[165,295],[165,285],fill="white")

canvasrealtime.create_polygon([142,265],[158,265],[158,290],[142,290],[142,265],fill="yellow")

        x6.set("This right turn left")
        polyleft = canvasrealtime.create_polygon([50, 50, 80, 20, 80, 80, 50, 50],
outline='red', fill='red',
                                width=2)  # triangle left
        canvasrealtime.create_line(150-(15*disty2), 0, 150-(15*disty2), 300, fill="black")
        canvasrealtime.create_line(150-(15*disty2),150,150,150,fill="red",width=5)
        canvasrealtime.create_line(150,140,150,160,fill="red",width=5)
        canvasrealtime.create_line(150-(15*disty2), 140, 150-(15*disty2), 160, fill="red",
width=5)
        canvasrealtime.create_text(150-(15*disty2)+20, 150, fill="darkblue", font="Times
20 italic bold",
                        text=disty2)

    if (angle_track > angle_path) : #and (disty2 > 1)
        canvasrealtime.delete("all")
        canvasrealtime.create_line(150+(15*disty2), 0, 150+(15*disty2), 300, fill="peru",
width=50)

canvasrealtime.create_polygon([140,240],[160,240],[163,255],[137,255],[140,240],fill="red")
```

```
canvasrealtime.create_polygon([135,255],[165,255],[165,295],[135,295],[135,255],fill="bl
ue")

canvasrealtime.create_polygon([130,255],[135,255],[135,265],[130,265],[130,255],fill="w
hite")

canvasrealtime.create_polygon([130,285],[135,285],[135,295],[130,295],[130,285],fill="w
hite")

canvasrealtime.create_polygon([165,255],[170,255],[170,265],[165,265],[165,255],fill="w
hite")

canvasrealtime.create_polygon([165,285],[170,285],[170,295],[165,295],[165,285],fill="w
hite")

canvasrealtime.create_polygon([142,265],[158,265],[158,290],[142,290],[142,265],fill="ye
llow")

        x6.set("This left turn right")
        polyright = canvasrealtime.create_polygon([250, 50, 220, 20, 220, 80, 250, 50],
outline='red', fill='red',
                                width=2)  # triangle right
        canvasrealtime.create_line(150+(15*disty2), 0, 150+(15*disty2), 300, fill="black")
        canvasrealtime.create_line(150+(15*disty2),150,150,150,fill="red",width=5)
        canvasrealtime.create_line(150,140,150,160,fill="red",width=5)
        canvasrealtime.create_line(150+(15*disty2), 140, 150+(15*disty2), 160, fill="red",
width=5)
        canvasrealtime.create_text(150+(15*disty2)-20, 150, fill="darkblue", font="Times
20 italic bold",
                           text=disty2)
```

```
    if disty2 <= 0.5:
        canvasrealtime.delete('all')
        canvasrealtime.create_line(150, 0, 150, 300, fill="peru", width=50)
        canvasrealtime.create_line(150,0,150,300,fill='black',width=50) #triangle right

canvasrealtime.create_polygon([140,240],[160,240],[163,255],[137,255],[140,240],fill="re
d")

canvasrealtime.create_polygon([135,255],[165,255],[165,295],[135,295],[135,255],fill="bl
ue")

canvasrealtime.create_polygon([130,255],[135,255],[135,265],[130,265],[130,255],fill="w
hite")

canvasrealtime.create_polygon([130,285],[135,285],[135,295],[130,295],[130,285],fill="w
hite")

canvasrealtime.create_polygon([165,255],[170,255],[170,265],[165,265],[165,255],fill="w
hite")

canvasrealtime.create_polygon([165,285],[170,285],[170,295],[165,295],[165,285],fill="w
hite")

canvasrealtime.create_polygon([142,265],[158,265],[158,290],[142,290],[142,265],fill="ye
llow")
        canvasrealtime.create_polygon([150,20,180,50,120,50,150,20],
outline='red',fill='red',width=2)
# calculate distance between path and rover , side of approach to path and display
in GUI

def myvar(*args):
```

```
    global lat, lng, latprelast, lngprelast, dist,q,hi,ratio,fl,distance,sidecar,duration, latch,
lngch, latsh, lngsh
    distance = 0
    sidecar = ""
    timer = str(time.strftime('%Y/%m/%d %H:%M:%S', time.gmtime()))
    data = []
    data = s.recv(1024)  # defualt is 1024
    ######## the data that brint to calculate math.st not be emath.ty
##############
    if len(data) == 129:

        word = data.split()

        duration = str(word[0:1])
        lat = float(word[2])
        lng = float(word[3])
        #lat = 16.59625733
        #lng = 100.41018024
        hi = float(word[4])
        q = int(word[5])
        sat = int(word[6])
        sde = float(word[7])
        sdn = float(word[8])
        sdu = float(word[9])
        sden = float(word[10])
        sdnu = float(word[11])
        sdeu = float(word[12])
        age = float(word[13])
        ratio = float(word[14])
    lat = float(lat)
    lng = float(lng)
```

```
    altitude = hi
    qq = q
    lngsh = ('%.8f' % lng)
    latsh = ('%.8f' % lat)
    numline = sum(1 for line in open("waysingle.txt"))
    numline = int(numline)
    x9.set(numline)
    numline2 = sum(1 for line in open("way4point.txt"))
    numline2= int(numline2)
    x10.set(numline2)

    if q == 1:
        qqq = "Fixed"
        latfix = lat
        lngfix = lng
    if q == 2:
        qqq = "Float"
        latfloat = lat
        lngfloat = lng
    if q == 5:
        qqq = "Single"
    try:#if (h+c+g)>max(h,c,g):
        ts = time.time()
        fl = 0
        predicroll, predicpitch, yaw = readtilt()
        wire = entrywire.get()
        wire = float(wire)
        latc,lngc,altc = compensate(lat,lng,hi,predicroll, predicpitch, yaw, wire) # Choose
Height Here!
        latch = ('%.8f' % latc)
        lngch = ('%.8f' % lngc)
```

```
        try:
          thepath = readline(latch, lngch)
          [distance,sidecar] = calforsave(float(latch),
float(lngch),float(thepath[0]),float(thepath[1]),float(thepath[2]),float(thepath[3]))
          Duration = str(duration)
          Sidecar = str(sidecar)
          Distance = float(distance)
          savesol(timer,float(latch),float(lngch),float(altitude),ratio,Distance,Sidecar)
          bufferauto(Distance,Sidecar)
        except:
          pass
        x1.set(timer)
        x2.set(latch)
        x3.set(lngch)
        x4.set(altc)
        x5.set(qqq)
        x8.set("Sensor On")
        bufferdraw(latch,lngch,qq)
        draw(lat,lng,q)
        Duration = str(duration)
        Sidecar = str(sidecar)
        Distance = float(distance)
        bufferauto(Distance,Sidecar)
        savesol(timer,float(latch),float(lngch),float(altitude),ratio,Distance,Sidecar)
        bufferauto(Distance,Sidecar)

    except TypeError:
      try:
        thepath = readline(latsh, lngsh)
        [distance,sidecar] = calforsave(float(latsh),
float(lngsh),float(thepath[0]),float(thepath[1]),float(thepath[2]),float(thepath[3]))
```

```
            Duration = str(duration)
            Sidecar = str(sidecar)
            Distance = float(distance)
            savesol(timer,float(latsh),float(lngsh),float(altitude),ratio,Distance,Sidecar)
            bufferauto(Distance,Sidecar)
        except:
            pass
        x1.set(timer)
        x2.set(latsh)
        x3.set(lngsh)
        x4.set(hi)
        x5.set(qqq)
        x8.set("Sensor Off")
        bufferdraw(latsh,lngsh,qq)
        yawt = 0
        draw(lat,lng,q)
        Duration = str(duration)
        Sidecar = str(sidecar)
        Distance = float(distance)
        savesol(timer,float(latsh),float(lngsh),float(altitude),ratio,Distance,Sidecar)
        bufferauto(Distance,Sidecar)

try:
    if checkautocar != None :
        autocar2()
except:
    pass
```

```
    return
# read raw position data and show in gui

######################## Pomath.y zone
###################################

######################## Tkinter zone ######################
def scanning(*args):
    if running:  # Only do this if the Stop button has not been clicked
        myvar()
    # After 1 second, call scanning again (create a recursive loop)
    root.after(100, scanning)
# scan GUI on or off

def start():
    """Run RTKRCV and Enable scanning by setting the global flag to True."""

    os.system("sh /home/pi/Desktop/program/rtkrcv.sh")
    global s
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    host = socket.gethostbyname(socket.gethostname())
    port = 52001
    time.sleep(5)
    s.connect((host, port))
    global running
    running = True
# start GUI

def stop():
    """Stop scanning by setting the global flag to False."""
```

```
    global running
    running = False
    os.system("sudo killall -9 python rtkrcv")
    stopcar2()
    print("Program has stop!")
# stop GUI

def default():
    print("default")
    subprocess.call("reset.sh")
# reset base station

def singlemodeconf():
    os.system("sh /home/pi/Desktop/program/single.sh")
# select standalone rover

def mybase():
    os.system("sh /home/pi/Desktop/program/mybase.sh")
# select pair base
def e12conf():
    os.system("sh /home/pi/Desktop/program/e12base.sh")
# select E12base
def save():
    myvar1 = aa.get()  # position mode
    print(myvar1)
    myvar2 = bb.get()  # type
    print(myvar2)
    myvar3 = cc.get()  # base position
    print(myvar3)
    myvar4 = dd.get()  # sol
    print(myvar4)
```

```
    myvar5 = ee.get()  # sol format
    print(myvar5)
    parser.set('i/oSection', 'pos1-posmode', myvar1)
    parser.set('i/oSection', 'inpstr2-type', myvar2)
    parser.set('i/oSection', 'ant2-postype', myvar3)
    parser.set('i/oSection', 'outstr2-type', myvar4)
    parser.set('i/oSection', 'outstr2-format', myvar12)
    with open('/home/pi/Desktop/program/rtkrcv.conf', 'w') as configfile:
        parser.write(configfile)
    print("Done")
# save using config file

def dolpsn():
    os.system("sh /home/pi/Desktop/program/dol.sh")
# select DOL base

def numup():
    global page
    page += 2
    while page >= 9:
        page = 1
    return page
# zoom variable 4 step

def draw(lat,lng,q):
    canvas.delete("all")#   ca                              2
    canvas2.delete("all")
    scale = page
    canvas2.create_oval(95,95,105,105,fill="yellow")#circle yellow

    try:
```

```
f = open("/home/pi/Desktop/program/way.txt")
way1ref = f.readline()
latway1,lngway1 = way1ref.split(' ')
latway1 = float(latway1)
lngway1 = float(lngway1)
utm_myorig = utm.from_latlon(latway1,lngway1)  # convert origin point to utm
X_orig = utm_myorig[0]  # East vary
Y_orig = utm_myorig[1]  # North vary
realpath = readline(lat,lng)
utmrealstart = utm.from_latlon(float(realpath[0]),float(realpath[1]))
X_realstart = utmrealstart[0]
Y_realstart = utmrealstart[1]
utmrealend = utm.from_latlon(float(realpath[2]),float(realpath[3]))
X_realend = utmrealend[0]
Y_realend = utmrealend[1]
dX_realstart = float(X_realstart-X_orig)
dY_realstart = float(Y_realstart-Y_orig)
dX_realend = float(X_realend-X_orig)
dY_realend = float(Y_realend-Y_orig)
utm_mycar = utm.from_latlon(lat,lng)  # convert origin point to utm
X_car = utm_mycar[0]
Y_car = utm_mycar[1]
dX_car = scale*(X_car-X_orig)
dX_car = float(dX_car)
dY_car = scale*(Y_car-Y_orig)
dY_car = float(dY_car)
points = []
biasx0=[]
biasy0=[]
diasx0=[]
diasy0=[]
```

```
with open("way2canvas.txt", "r") as infile:
    for line in infile:
        line = line.split()
        x, y = (float(line[2]),-1*float(line[3]))
        points.append((x,y))
# find the max dimension
for i in range(0, len(points)):
    biasx   =abs(points[i][0])
    biasy   =abs(points[i][1])
    biasx0.append(biasx)
    biasy0.append(biasy)
for i in range(0, len(points)):
    diasx = points[i][0]
    diasy = points[i][1]
    diasx0.append(diasx)
    diasy0.append(diasy)
diasxmax = max(diasx0) #add new 1902
diasymax = max(diasy0)
diasxmin = min(diasx0)
diasymin = min(diasy0)
# Check maximum length of x and y axis
biasxmax = max(biasx0) #add new 1902
biasymax = max(biasy0)
biasxmin = min(biasx0)
biasymin = min(biasy0)
cenmax = max(biasxmax,biasymax)
cenx = ((biasxmax+biasxmin)/(2*biasxmax))*150
ceny = 300-(((biasymax+biasymin)/(2*biasymax))*150)
centerscale = 100/cenmax #100 canvas = scalemax
centerscale1 = 100/cenmax
centerscale2 = 50/cenmax
```

```
absx = []
absy = []
for biasxi in biasx0:
    absbiasxi = abs(biasxi)
    absx.append(absbiasxi)
for biasyi in biasx0:
    absbiasyi = abs(biasyi)
    absy.append(absbiasyi)
maxx = max(biasx0)
maxy = max(biasy0)
minx = min(biasx0)
miny = min(biasy0)
xx = (maxx+minx)/2
yy = (maxy+miny)/2
biasx = (200/2)
biasy = (200/2)
biasx1 = biasx
biasx2 = 150
biasy2 = 150
# Check type of layout mode
checksingle = open("/home/pi/Desktop/program/waysingle.txt")
singlepath = len(checksingle.readlines())
checkmuti = open("/home/pi/Desktop/program/way4point.txt")
multipath = len(checkmuti.readlines())
if (singlepath > 0) and (multipath == 0):
    dd = 1
elif (singlepath == 0) and (multipath > 0):
    dd=2
totalcanvasx=[]
totalcanvasy=[]
```

```
        for i in range(0,len(points)-1,dd): # old is "for i in range(0,len(points)-1):" if you
have to use with multiwaypoint you will use "for i in range(0,len(points)-1,2):" instead.
            canvas.create_line((centerscale2*scale * points[i][0] )+
biasx2,(centerscale2*scale * points[i][1] )+ biasy2,(centerscale2*scale * points[i +
1][0])+ biasx2,(centerscale2*scale*points[i + 1][1])+ biasy2, fill="black",width = 3)

            canvas2.create_line((centerscale*scale * points[i][0] )+
biasx1+dX_car,(centerscale*scale * points[i][1] )+ biasy1+dY_car,(centerscale*scale *
points[i + 1][0])+ biasx1+dX_car,(centerscale*scale*points[i + 1][1])+ biasy1+dY_car,
fill="black",width = 3)

            canvas2.create_line(centerscale*scale*dX_realstart + biasx1+dX_car, -
1*centerscale*scale * dY_realstart + biasy1+dY_car, centerscale*scale * dX_realend +
biasx1+dX_car,-1*centerscale1*scale * dY_realend + biasy1+dY_car, fill="white",width
= 2)

        canvas.create_line(250,280,250,300,fill='black')
        canvas.create_line(300,280,300,300,fill='black')
        canvas.create_line(150,290,300,290,fill='black')
        canvas.create_text(260,285, fill="darkblue", font="Times 10 italic bold",
                text=cenmax)


    except:
        pass
# draw canvas grapphic
def Save_Route(event=None):
    root.title("save to file Route.png")
    time.sleep(2)
    root.title("Complete")
    filename = "Route.png"
    img_pil.save(filename)
```

```
    image = Image.open(filename)
    image.show()
# save picture path
def createway():
    f = open('way4point.txt','a+')
    f.truncate(0)
    f.close
    f = open('waysingle.txt','a+')
    lat,lng,Q = readbufferdraw()
    latfix = float(lat)
    lngfix = float(lng)
    f.write('%.8f %.8f\n' % (latfix, lngfix))
    f.close
# create way
def deleteway():
    readFile = open("waysingle.txt")
    lines = readFile.readlines()
    readFile.close()
    w = open("waysingle.txt", 'w')
    w.writelines([item for item in lines[:-1]])
    w.close()
# delete way

def deleteall():
    f = open('waysingle.txt','a+')
    f.truncate(0)  # need '0' when using r+
    f.close
    f = open('way2canvas.txt','a+')
    f.truncate(0)  # need '0' when using r+
    f.close
# delete way
```

```
def checkmode(*args):
    try:
        ft = float(trap)
        d = 2
    except TypeError:
        d = 1
    return d
def createway4pointdualcorner():
    f = open('waysingle.txt','a+')
    f.truncate(0)
    f.close
    f = open('way4point.txt','r')
    file = f.readlines()
    L = (len(file))
    f.close
    if L < 4 :
        f = open('way4point.txt', 'a+')
        lat, lng, Q = readbufferdraw()
        latfix = lat
        lngfix = lng
        f.write('%.8f %.8f\n' % (latfix, lngfix))
        f.close
# check waypoint mode

def submitsinglemode():
    os.system("cp /home/pi/Desktop/program/waysingle.txt
/home/pi/Desktop/program/way.txt")
# confirm waypoint

def deleteway4point():
```

```
    readFile = open("way4point.txt")
    lines = readFile.readlines()
    readFile.close()
    w = open("way4point.txt", 'w')
    w.writelines([item for item in lines[:-1]])
    w.close()
# create corner of 4 point

def deleteallway4point():
    f = open('way4point.txt','a+')
    f.truncate(0)  # need '0' when using r+
    f.close
    f = open('way4pointbuffer.txt','a+')
    f.truncate(0)
    f.close
    f = open('way2canvas.txt','a+')
    f.truncate(0)  # need '0' when using r+
    f.close
# delete way

def createbreakpoint():
    f = open('way.txt', 'a+')
    f.truncate(0)
    f.close()
    f = open("way4point.txt", 'r', encoding='utf-8')
    wayone = f.readline()
    waytwo = f.readline()
    waythree = f.readline()
    wayfour = f.readline() # 1 duo 3 , 2 duo 4
    w1 = wayone.split(' ')
    lat1 = float(w1[0])
```

```
lng1 = float(w1[1])
w2 = waytwo.split(' ')
lat2 = float(w2[0])
lng2 = float(w2[1])
w3 = waythree.split(' ')
lat3 = float(w3[0])
lng3 = float(w3[1])
w4 = wayfour.split(' ')
lat4 = float(w4[0])
lng4 = float(w4[1])
utmpoint1 = utm.from_latlon(lat1,lng1)  # convert origin point to utm
X1 = utmpoint1[0]  # East vary
Y1 = utmpoint1[1]  # North vary
para1 = utmpoint1[2]
para2 = utmpoint1[3]
utmpoint2 = utm.from_latlon(lat2, lng2)
X2 = utmpoint2[0]
Y2 = utmpoint2[1]
utmpoint3 = utm.from_latlon(lat3, lng3)
X3 = utmpoint3[0]
Y3 = utmpoint3[1]
utmpoint4 = utm.from_latlon(lat4, lng4)
X4 = utmpoint4[0]
Y4 = utmpoint4[1]
Dx_line13 = X3 - X1
Dy_line13 = Y3 - Y1
signline13X = math.copysign(1,Dx_line13)
signline13Y = math.copysign(1,Dy_line13)
anglepoint13 = math.degrees(math.atan2(float(Dy_line13), float(Dx_line13)))-90
Dx_line24 = X4 - X2
Dy_line24 = Y4 - Y2
```

```
signline24X = math.copysign(1, Dx_line24)
signline24Y = math.copysign(1, Dy_line24)
anglepoint24 = math.degrees(math.atan2(float(Dy_line24), float(Dx_line24)))-90
Dispoint13 = Dx_line13 + Dy_line13
Dispoint24 = Dx_line24 + Dy_line24
for i in range(0,4):
    f = open('way4point.txt', 'r')
    wayref1 = f.readline()
    wayref2 = f.readline()
    wayref3 = f.readline()
    wayref4 = f.readline()
    f.close()
f = open('way.txt', 'a+')
f.seek(0, 0)
f.write('%s%s' % (wayref2,wayref1))
f.close()
n = entry.get()
n = int(n)
latlonfromutm13 = []
latlonfromutm24 = []
way1 = []
way2 = []
way = []
way11 = []
way111 = []
way22 = []
way222 = []
way.append([])
latlon241 = []
for i in range(1,n+1):
    breakpoint13 = (i * (Dispoint13 / (n + 1)))
```

```
        breakpointX13 =
abs((i*(Dispoint13/(n+1)))*math.sin(math.radians(anglepoint13)))*signline13X
        breakpointY13 =
abs((i*(Dispoint13/(n+1)))*math.cos(math.radians(anglepoint13)))*signline13Y
        utm13X = utmpoint1[0]  + breakpointX13
        utm13Y = utmpoint1[1] + breakpointY13
        latlon13 = utm.to_latlon(utm13X,utm13Y,para1,para2)
        latlonfromutm13.append(latlon13)
        breakpoint24 = (i * (Dispoint24 / (n + 1)))
        breakpointX24 = abs((i * (Dispoint24 / (n + 1))) *
math.sin(math.radians(anglepoint24)))*signline24X
        breakpointY24 = abs((i * (Dispoint24 / (n + 1))) *
math.cos(math.radians(anglepoint24)))*signline24Y
        utm24X = utmpoint2[0] + breakpointX24
        utm24Y = utmpoint2[1] + breakpointY24
        latlon24 = utm.to_latlon(utm24X, utm24Y, para1, para2)
        latlonfromutm24.append(latlon24)
    way0 = latlonfromutm13[0]
    way[0].append(way0)
    f = open('way4pointbuffer.txt', 'a+')
    f.truncate(0)
    way[0].append(way0)

    for i in range(0,n-1):
        way0 = latlonfromutm13[i]
        way11 = latlonfromutm13[i]
        way111 = latlonfromutm13[i+1]
        way22 = latlonfromutm24[i]
        way222 = latlonfromutm24[i+1]
        if ( i %2 == 0):
            f.write("%s\n%s\n%s\n%s\n" % (way11, way22, way222, way111))
```

```
        elif(i >= n-2 ):
            f.write("%s\n%s\n" % (latlonfromutm13[n-1],latlonfromutm24[n-1]))
    for i in range(0, (2*n)):
        f = open('way4pointbuffer.txt', 'r')
        f1 = open('way.txt', 'a+')
        file = f.readlines()
        x = file[i]
        z, u = x.split(',')
        z = z.replace("(", "")
        u = u.replace(")", "")
        u = u.replace(" ", "")

        f1.write('%s %s' % (z, u))
        f.close()
        f1.close()
    f = open('way.txt', 'a+')
    f.seek(0, 0)
    f.write('%s%s' % (wayref3,wayref4))
    f.close()
# delete way

def createnewbreakpoint():
    global trap
    f = open('way.txt', 'a+')
    f.truncate(0)
    f.close()
    f = open("way4point.txt", 'r', encoding='utf-8')
    wayone = f.readline()
    waytwo = f.readline()
    waythree = f.readline()
    wayfour = f.readline() # 1 duo 3 , 2 duo 4
```

```
w1 = wayone.split(' ')
lat1 = float(w1[0])
lng1 = float(w1[1])
w2 = waytwo.split(' ')
lat2 = float(w2[0])
lng2 = float(w2[1])
w3 = waythree.split(' ')
lat3 = float(w3[0])
lng3 = float(w3[1])
w4 = wayfour.split(' ')
lat4 = float(w4[0])
lng4 = float(w4[1])
# find utm coordinate
utmpoint1 = utm.from_latlon(lat1,lng1)  # convert origin point to utm
X1 = utmpoint1[0]  # East vary
Y1 = utmpoint1[1]  # North vary
para1 = utmpoint1[2]
para2 = utmpoint1[3]
utmpoint2 = utm.from_latlon(lat2, lng2)
X2 = utmpoint2[0]
Y2 = utmpoint2[1]
utmpoint3 = utm.from_latlon(lat3, lng3)
X3 = utmpoint3[0]
Y3 = utmpoint3[1]
utmpoint4 = utm.from_latlon(lat4, lng4)
X4 = utmpoint4[0]
Y4 = utmpoint4[1]
# move origin to X1,Y1
XY1 = np.array([X1,Y1])
XY2 = np.array([X2,Y2])
XY3 = np.array([X3,Y3])
```

```
XY4 = np.array([X4,Y4])
v0 = XY1-XY1
v1 = XY2-XY1
v2 = XY3-XY1
v3 = XY4-XY1
v01 = v1-v0
v02 = v2-v0
v13 = v3-v1
v03 = v3-v0
v12 = v2-v1
theta1 = np.degrees(np.arctan2(v03[1],v03[0]))
theta2 = np.degrees(np.arctan2(v12[1],v12[0]))
L1 = np.sqrt((v02[0]**2)+(v02[1]**2))
L2 = np.sqrt((v13[0]**2)+(v13[1]**2))
# add width
width = max(np.sqrt((v03[0]**2)+(v03[1]**2)),np.sqrt((v12[0]**2)+(v12[1]**2)))
for i in range(0,4):
    f = open('way4point.txt', 'r')
    wayref1 = f.readline()
    wayref2 = f.readline()
    wayref3 = f.readline()
    wayref4 = f.readline()
    f.close()
# stock end pos
wayref3stock = wayref3
wayref4stock = wayref4
f = open('way.txt', 'a+')
f.seek(0, 0)
f.write('%s%s' % (wayref1,wayref2))#f.write('%s%s' % (wayref2,wayref1))
f.close()
trap = entry.get()
```

```
n = int(round(width/float(trap)))
latlonfromutm13 = []
latlonfromutm24 = []
way1 = []
way2 = []
way = []
way11 = []
way111 = []
way22 = []
way222 = []
way.append([])
latlon241 = []
for i in range(1,n+1):
    breakpointX13 = abs((i*(float(trap)))*math.cos(math.radians(theta1)))
    breakpointY13 = abs((i*(float(trap)))*math.sin(math.radians(theta1)))
    utm13X = X1 + v0[0] + breakpointX13
    utm13Y = Y1 + v0[1] + breakpointY13
    latlon13 = utm.to_latlon(utm13X,utm13Y,para1,para2)
    latlonfromutm13.append(latlon13)
    breakpointX24 = abs((i * (float(trap))) * math.cos(math.radians(theta2)))
    breakpointY24 = abs((i * (float(trap))) * math.sin(math.radians(theta2)))
    utm24X = X2 +breakpointX24
    utm24Y = Y2 +breakpointY24
    latlon24 = utm.to_latlon(utm24X, utm24Y, para1, para2)
    latlonfromutm24.append(latlon24)
print("utm13", latlonfromutm13)
print("utm24", latlonfromutm24)
way0 = latlonfromutm13[0]
way[0].append(way0)
f = open('way4pointbuffer.txt', 'a+')
f.truncate(0)
```

```
    way[0].append(way0)
    for i in range(0,n-1):
        way0 = latlonfromutm13[i]
        way11 = latlonfromutm13[i]
        way111 = latlonfromutm13[i+1]
        way22 = latlonfromutm24[i]
        way222 = latlonfromutm24[i+1]
        if ( i %2 == 0):
            f.write("%s\n%s\n%s\n%s\n" % (way22, way11, way111,
way222))#f.write("%s\n%s\n%s\n%s\n" % (way11, way22, way222, way111))#
        elif(i >= n-2 ):
            f.write("%s\n%s\n" % (latlonfromutm24[n-1],latlonfromutm13[n-
1]))#f.write("%s\n%s\n" % (latlonfromutm13[n-1],latlonfromutm24[n-1]))#
    for i in range(0, (2*n)):
        f = open('way4pointbuffer.txt', 'r')
        f1 = open('way.txt', 'a+')
        file = f.readlines()
        x = file[i]
        z, u = x.split(',')
        z = z.replace("(", "")
        u = u.replace(")", "")
        u = u.replace(" ", "")
        f1.write('%s %s' % (z, u))
        f.close()
        f1.close()
    # check the last position from way4pointbuffer.txt

    return trap
# calculate position of path

def way2canvas():
```

```
global lats,lngs,orig,latw,lngw
with open("way.txt", "r") as f, open('way2canvas.txt', 'r+') as outfile:
    file = f.readlines()
    L = round(len(file))
    ws = file[0].split(' ')
    lats = float(ws[0])
    lngs = float(ws[1])
    ws2 = [lats,lngs]
    utmorig = utm.from_latlon(lats, lngs) # convert origin point to utm
    X_orig = utmorig[0] # East vary
    Y_orig = utmorig[1] # North vary
    dX_orig = 0 # X origin
    dY_orig = 0 # Y origin
    path = []
    path.append([lats,lngs,dX_orig,dY_orig])
    outfile.truncate(0)  # need '0' when using r+
    outfile.write("%.8f %.8f %.4f %.4f %i\n" %(lats,lngs,dX_orig,dY_orig,1))
# Choose the first point to be the origin
    for w in range(1,L):
        wp = file[w].split(' ')
        latw = float(wp[0])
        lngw = float(wp[1])
        utmw = utm.from_latlon(latw, lngw) # convert tracking point to utm
        X_track = utmw[0] # East vary
        Y_track = utmw[1] #11111111111111111.. North vary
        dX_Track = X_track-X_orig
        dY_Track = Y_track-Y_orig
        path.append([latw,lngw,dX_Track,dY_Track])
        outfile.write("%.8f %.8f %.4f %.4f %i\n" %(latw,lngw,dX_Track,dY_Track,w+1))
os.system("python3 /home/pi/Desktop/program/pygmaps.py")
return path #file
```

```
# calculate xy of canvas from raw position data

def openmymap():
    os.system("xdg-open /home/pi/Desktop/program/mymap.html")
# open google map

def exitprogram(*args):
    stopcar()
    if h != None:
        pi.i2c_close(h)
        pi.i2c_close(c)
        pi.i2c_close(g)
        pi.stop()
    root.quit()
# exit GUI

######################## car motor ############################
def stopcar():
    gpio.output(11, False)
    gpio.output(13, False)
    gpio.output(15, False)
    gpio.output(19, False)
    gpio.output(29, False)
    gpio.output(31, False)
    mypwm1.stop()
    mypwm2.stop()
    gpio.cleanup()
# stop autosteer car

def stopcar2():
    mypwm1.ChangeDutyCycle(0)
```

```
    mypwm2.ChangeDutyCycle(0)
    gpio.output(11, False)
    gpio.output(13, False)
    gpio.output(15, False)
    gpio.output(19, False)
 # stop autosteer car

def forward():
    mypwm1.ChangeDutyCycle(70)
    mypwm2.ChangeDutyCycle(70)
    gpio.output(11, True)
    gpio.output(13, False)
    gpio.output(15, True)
    gpio.output(19, False)
 # forward autosteer car

def reverse():
    mypwm1.ChangeDutyCycle(70)
    mypwm2.ChangeDutyCycle(70)
    gpio.output(11, False)
    gpio.output(13, True)
    gpio.output(15, False)
    gpio.output(19, True)
 # reverse autosteer car

def turn_leftbackright(idL):


   if idL == 0 or idL == 1 :
       print("<<<<<<<<<<",idL)
```

```
        mypwm1.ChangeDutyCycle(60)
        mypwm2.ChangeDutyCycle(80)
        gpio.output(11, True)
        gpio.output(13, False)
        gpio.output(15, True)
        gpio.output(19, False)
    elif idL == 2 :
        backright()

 # turn left autosteer car


def backright():
    print("RRRRRRRRRRR")
    mypwm1.ChangeDutyCycle(70)
    mypwm2.ChangeDutyCycle(60)
    gpio.output(11, True)
    gpio.output(13, False)
    gpio.output(15, True)
    gpio.output(19, False)
# turn right autosteer car
def turn_rightbackleft(idR):


    if idR == 0 or idR == 1 :
        print(">>>>>>>>>>>>>",idR)
        mypwm1.ChangeDutyCycle(80)
        mypwm2.ChangeDutyCycle(60)
        gpio.output(11, True)
        gpio.output(13, False)
        gpio.output(15, True)
```

```
        gpio.output(19, False)
    elif idR == 2 :
        backleft()

# turn right autosteer car

def backleft():
    print("LLLLLLLLLLL")
    mypwm1.ChangeDutyCycle(60)
    mypwm2.ChangeDutyCycle(70)
    gpio.output(11, True)
    gpio.output(13, False)
    gpio.output(15, True)
    gpio.output(19, False)
# turn left autosteer car
def autocar():
    global checkautocar
    checkautocar = 1
# global variable to use autosteer

def autocar2():
    global turnbacklr
    #print(turnbacklr)
    turnbacklr = turnbacklr + 1
    if turnbacklr > 2 :
        turnbacklr = 0

    print("Runnnnnnnnnnnnnnnnnnn")

    fileoo = open("autobuffer.txt", "r+")
    data = (fileoo.read()).split(" ")
```

```
    disty = data[0]
    disty = float(disty)
    side = data[1]
    if disty <= 0.08:
        forward()
    elif (disty <= 5) and (side == "turn_left"):
        #print("<<<<<<----------")
        turn_leftbackright(turnbacklr)
    elif (disty <= 5) and (side == "turn_right"):

        #print("---------->>>>>>>")

        turn_rightbackleft(turnbacklr)

    elif (disty > 5) :
        stopcar2()
        print("----------")
# variable to control autosteer

##################### Tkinter zone ##############################
root = tk.Tk()

x1 = IntVar() #date
x2 = IntVar() #lat
x3 = IntVar() #lng
x4 = IntVar() #height
x5 = StringVar() #quality
x6 = StringVar() #turn
x7 = IntVar() #disy
x8 = StringVar() # Sensor status
x9 = IntVar() #numline
```

```
x10 = IntVar() #numline2----> 4point
# use width x height + x_offset + y_offset (no spaces!)
root.geometry('600x450')  # root.geometry("%dx%d+%d+%d" % (300, 200, 100, 50))
root.title('KMITL Agricultural 4.0')

nb = ttk.Notebook(root)
nb.pack(fill='both', expand='yes')

# create a child frame for each page
f1 = tk.Frame(bg='mint cream')
#f2 = tk.Frame(bg='light goldenrod')
f3 = tk.Frame(bg='gainsboro')
f4 = tk.Frame(bg='gainsboro')
# create the pages
nb.add(f1, text='Settings')

nb.add(f3, text='Display Map')
nb.add(f4, text='Display Path')

# put a button widget on child frame f1 on page1
# Page1

frame1 = Frame(f1)
frame1.pack()
frame01 = LabelFrame(f1,text="Base station", highlightbackground="black",
highlightcolor="black", highlightthickness=1, width=500, height=250, bd= 0)
frame01.place(relx=0.075 ,rely=0.05)
frame02 = LabelFrame(f1,text="Sensor", highlightbackground="black",
highlightcolor="black", highlightthickness=1, width=500, height=100, bd= 0)
frame02.place(relx=0.075 ,rely=0.7)
sp1 = Button(frame01, text="DOL", command=dolpsn)
```

```
sp1.place(relx=0.785,rely=0.1,relwidth=0.18)
detail1 = Label(frame01,text="detail DOL")
detail1.place(relx=0.785,rely=0.25)
sp2 = Button(frame01, text="My Base", command=mybase)
sp2.place(relx=0.535,rely=0.1,relwidth=0.18)
detail2 = Label(frame01,text="My pair base")
detail2.place(relx=0.535,rely=0.25)

sp3 = Button(frame01, text="E12 Base", command=e12conf)
sp3.place(relx=0.285,rely=0.1,relwidth=0.18)
detail = Label(frame01,text="KMITL E12 Base station")
detail.place(relx=0.285,rely=0.25)

save = tk.Button(frame01, text='Single mode', command=singlemodeconf)
save.place(relx=0.035,rely=0.1,relwidth=0.18)
variable = StringVar(root)
variable.set("single")  # default value
sensor = Label(frame02,text = "Antenna Height (m)")
sensor.place(relx=0.175,rely=0.1)

sep1 = ttk.Separator(frame01,orient="vertical")
sep1.place(relx=0.25,rely=0.1,relheight=0.5)
sep2 = ttk.Separator(frame01,orient="vertical")
sep2.place(relx=0.5,rely=0.1,relheight=0.5)
sep3 = ttk.Separator(frame01,orient="vertical")
sep3.place(relx=0.75,rely=0.1,relheight=0.5)
entrywire = Entry(frame02,text="wireheight")
entrywire.place(relx=0.45,rely=0.1)
calib = Button(frame02,text="calib",command=calibimu)
calib.place(relx=0.45, rely=0.5)
# Page3
```

```
a1 =Label(f3, width=30, textvariable=x1)
a2 =Label(f3, width=30, textvariable=x2)
a3 =Label(f3, width=30, textvariable=x3)
#a4 =Label(f3, width=30, textvariable=x4)
a5 =Label(f3, width=30, textvariable=x5)
a6 =Label(f3, width=30, textvariable=x6)
a7 =Label(f3, width=30, textvariable=x7)
a8 =Label(f3, width=30, textvariable=x8)

text1 = Label(f3,text = "Date")
text2 = Label(f3,text = "Latitude")
text3 = Label(f3,text = "Longtitude")
text4 = Label(f3,text = "Status")
text5 = Label(f3,text = "Move")
text6 = Label(f3,text = "Distance")
text8 = Label(f3,text = "Sensor Status")

text1.place(relx=0.00, rely=0.03, relheight=0.067, relwidth=0.176)
text2.place(relx=0.554, rely=0.05, relheight=0.067, relwidth=0.176)
text3.place(relx=0.554, rely=0.10, relheight=0.067, relwidth=0.176)
text4.place(relx=0.554, rely=0.807, relheight=0.067, relwidth=0.176)
text5.place(relx=0.554, rely=0.857, relheight=0.067, relwidth=0.176)
text6.place(relx=0.1, rely=0.9, relheight=0.067, relwidth=0.176)
text8.place(relx=0.554, rely=0.857, relheight=0.067, relwidth=0.176)

a1.place(relx=0.150, rely=0.03, relheight=0.067, relwidth=0.326)
a2.place(relx=0.724, rely=0.05, relheight=0.067, relwidth=0.226)
a3.place(relx=0.724, rely=0.10, relheight=0.067, relwidth=0.226)
a5.place(relx=0.724, rely=0.807, relheight=0.067, relwidth=0.226)
a6.place(relx=0.724, rely=0.857, relheight=0.067, relwidth=0.226)
a7.place(relx=0.2, rely=0.9, relheight=0.067, relwidth=0.226)
```

```
a8.place(relx=0.724, rely=0.857, relheight=0.067, relwidth=0.226)


btn6 = tk.Button(f3, text='zoom',command=numup)
btn6.place(relx = 0.95,rely = 0.22,relwidth=0.05)

autocarbutton = Button(f3 ,text="Autosteer",command=autocar,width=9)
autocarbutton.place(relx=0.1,rely=0.5,relheight=0.067, relwidth=0.306)
autocarbutton.pack()

canvasrealtime = Canvas(f3,width= width ,height = height,bg="yellowgreen")
canvasrealtime.place(relx=0 ,rely=0.1)
canvas2 = Canvas(f3,width = 200,height = 200,bg="yellowgreen")
canvas2.place(relx=0.55,rely=0.22)

canvasrealtime.create_line(150,0,150,300,fill="seashell3",width=50)

#page
a21=Label(f4, width=30, textvariable=x2)
a21.place(relx=0.25, rely=0.01, relheight=0.067, relwidth=0.276)
a31=Label(f4, width=30, textvariable=x3)
a31.place(relx=0.65, rely=0.01, relheight=0.067, relwidth=0.276)
a41 =Label(f4, width=30, textvariable=x5)
a41.place(relx=0.375, rely=0.9, relheight=0.067, relwidth=0.176)
a51=Label(f4, width=30, textvariable=x9)
a51.place(relx=0.8, rely=0.06, relheight=0.067, relwidth=0.176)
a61=Label(f4, width=30, textvariable=x10)
a61.place(relx=0.8, rely=0.450, relheight=0.067, relwidth=0.176)

text21 = Label(f4,text = "Latitude")
text21.place(relx=0.1, rely=0.01, relheight=0.067, relwidth=0.176)
```

```
text31 = Label(f4,text = "Longtitude")
text31.place(relx=0.5, rely=0.01, relheight=0.067, relwidth=0.176)
text41 = Label(f4,text = "Status")
text41.place(relx=0.225, rely=0.9, relheight=0.067, relwidth=0.176)
text51 = Label(f4,text = "Number of Way4point")
text51.place(relx=0.55, rely=0.450, relheight=0.067, relwidth=0.276)

canvas = Canvas(f4,width = 300,height = 300,bg="greenyellow")
canvas.place(relx=0.05 ,rely=0.085)
img_pil = Image.new("RGB",(width,height),color="lightgreen")
cv_pil = ImageDraw.Draw(img_pil)

openmap = Button(f4, text="GoogleMap", command=openmymap)
openmap.pack(side='left', anchor='sw', padx=3, pady=5)

frame2 = LabelFrame(f4,text="Single Waypoint", highlightbackground="black",
highlightcolor="black", highlightthickness=1, width=100, height=500, bd= 0)
frame2.place(relx=0.65 ,rely=0.125,relwidth=0.2)
frame3 = LabelFrame(f4,text="Multiple Waypoint", highlightbackground="black",
highlightcolor="black", highlightthickness=1, width=100, height=500, bd= 0)
frame3.place(relx=0.65,rely=0.500,relwidth=0.2)
save = Button(frame2,text="save",command=Save_Route)
save.pack()
createway = Button(frame2, text="Create point", command=createway,width=9)
createway.place(relx=0.2,rely=1,relheight=0.067, relwidth=0.176)
deleteway = Button(frame2, text="Delete point", command=deleteway,width=9)
deleteway.place(relx=0.2,rely=3,relheight=0.067, relwidth=0.176)
deleteall = Button(frame2, text="Delete all", command=deleteall,width=9)
deleteall.place(relx=0.2,rely=7,relheight=0.067, relwidth=0.176)
way2canvas1 = Button(f4, text="Show map", command=way2canvas,width=9)
submitsingle = Button(frame2, text="Submit",command=submitsinglemode,width=9)
```

```
submitsingle.place(relx=0.2,rely=5,relheight=0.067, relwidth=0.176)
way2canvas1.place(relx=0.665,rely=0.9)

entry = Entry(frame3,text="number",width=9)
entry.place(relx=0.1,rely=0.1)
createway4pointdualcorner = Button(frame3, text="Create point",
command=createway4pointdualcorner,width=9)
createway4pointdualcorner.place(relx=0.1,rely=0.2,relheight=0.067, relwidth=0.176)
deleteway4point = Button(frame3, text="Delete point",
command=deleteway4point,width=9)
deleteway4point.place(relx=0.1,rely=0.3,relheight=0.067, relwidth=0.176)
deleteallway4point = Button(frame3, text="Delete all",
command=deleteallway4point,width=9)
deleteallway4point.place(relx=0.1,rely=0.4,relheight=0.067, relwidth=0.176)
submit = Button(frame3 ,text="Submit", command=createnewbreakpoint,width=9)
submit.place(relx=0.1,rely=0.5,relheight=0.067, relwidth=0.176)
text61 = Label(f4,text = "Number of Waypoint")
text61.place(relx=0.55, rely=0.06, relheight=0.067, relwidth=0.276)

createway.pack()
deleteway.pack()
deleteall.pack()
submitsingle.pack()
entry.pack()
createway4pointdualcorner.pack()
deleteway4point.pack()
deleteallway4point.pack()
submit.pack()

start = Button(root, text="Start", command=start)
start.pack(side='left', anchor='sw', padx=3, pady=5)
```

```
root.after(100, scanning)  # After 1 second, call scanning
stop = Button(root, text="Stop", command=stop)
stop.pack(side='left', anchor='s', padx=3, pady=5)
Exit = Button(root, text="Exit", command=exitprogram)
Exit.pack(side='left', anchor='se', padx=3, pady=5)
root.mainloop()

# GUI graphic and button
```

ภาคผนวก ง

ชุดคำสั่งสำหรับการคำนวณตำแหน่งด้วยเทคนิค RTK

ชุดคำสั่ง e12base.sh สำหรับการเลือกใช้สถานีฐาน บนอาคาร 12 ชั้น

```
#!/bin/bash
rm rtkrcv.conf
cp  /home/pi/Desktop/program/rtkrcvE12.conf /home/pi/Desktop/program/rtkrcv.conf
echo E12Base
# config file E12
```

ชุดคำสั่ง mybase.sh สำหรับการเลือกใช้สถานีฐานส่วนบุคคล

```
#!/bin/bash
rm rtkrcv.conf
cp /home/pi/Desktop/program/mybase.conf /home/pi/Desktop/program/rtkrcv.conf
echo My Base
# config file pair base
```

ชุดคำสั่ง rtkrcv.sh สำหรับการใช้งานในโหมด Single

```
#!/bin/bash
rm rtkrcv.conf
cp /home/pi/Desktop/program/rtkrcvsingle.conf
/home/pi/Desktop/program/rtkrcv.conf
echo Single
# config file single mode
```

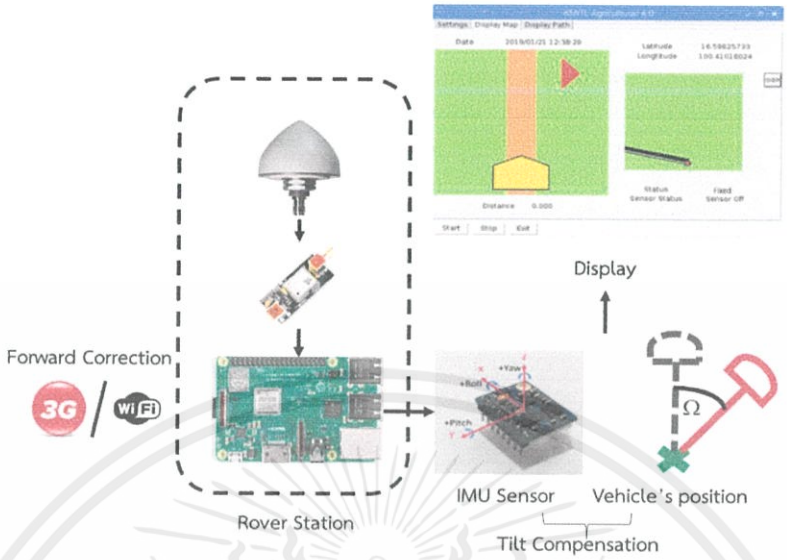
ภาคผนวก จ
คำสั่งสร้างฐานข้อมูล

```
#!/bin/bash
while IFS='' read -r line || [[ -n "$line" ]]; do
    echo "Text read from file: $line"
    echo "$content"
    STR=$line
    var1=$(echo $STR | cut -f1 -d,)
    var2=$(echo $STR | cut -f2 -d,)
    var3=$(echo $STR | cut -f3 -d,)
    var4=$(echo $STR | cut -f4 -d,)
    var5=$(echo $STR | cut -f5 -d,)
    var6=$(echo $STR | cut -f6 -d,)
    var7=$(echo $STR | cut -f7 -d,)
    var8=$(echo $STR | cut -f8 -d,)
    echo $var1
    echo $var2
    echo $var3
    echo $var4
    echo $var5
    echo $var6
    echo $var7
    echo $var8

    mysql -u TTS -pttsproj rtktrack -e "INSERT INTO rover(date,time,latitude
,longitude,height,ratio,distance,side)
VALUES('$var1','$var2','$var3','$var4','$var5','$var6','$var7','$var8');"

done < "$1"
# insert data to mysql table
```

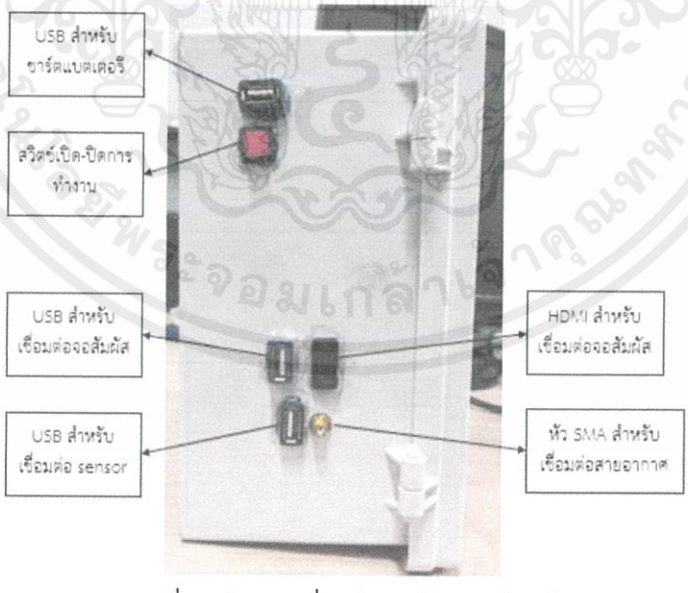
การติดตั้ง และใช้งานสถานีจลน์



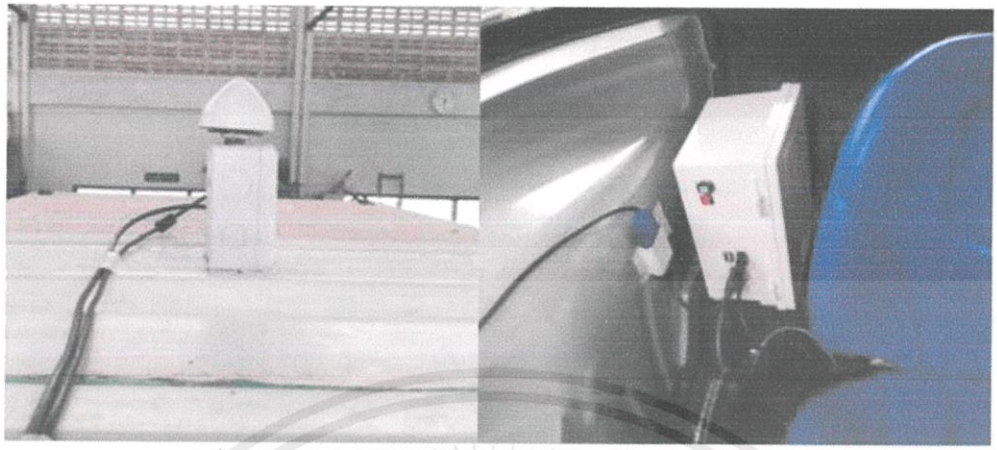
รูปที่ ๑ ภาพรวมของสถานีจลน์

ขั้นตอนการเชื่อมต่อตามรูปที่ ๑ มีดังนี้

๑. ต่อสายอากาศ TW-3740 เข้ากับหัว SMA ของสายนำสัญญาณด้าน และเชื่อมต่อสาย USB สำหรับเชื่อมต่อ SENSOR ตามรูปที่ ๒ แล้วทำการติดตั้งกล่องสายอากาศ และเซนเซอร์ IMU เข้ากับหลังคารถไถหรือรถเพาะปลูก จากนั้นวางกล่องหลักของสถานีจลน์ไว้ข้างเบาะผู้ขับขี่ ดังรูปที่ ๓



รูปที่ ๒ ช่องการเชื่อมต่อภายในสถานีจลน์



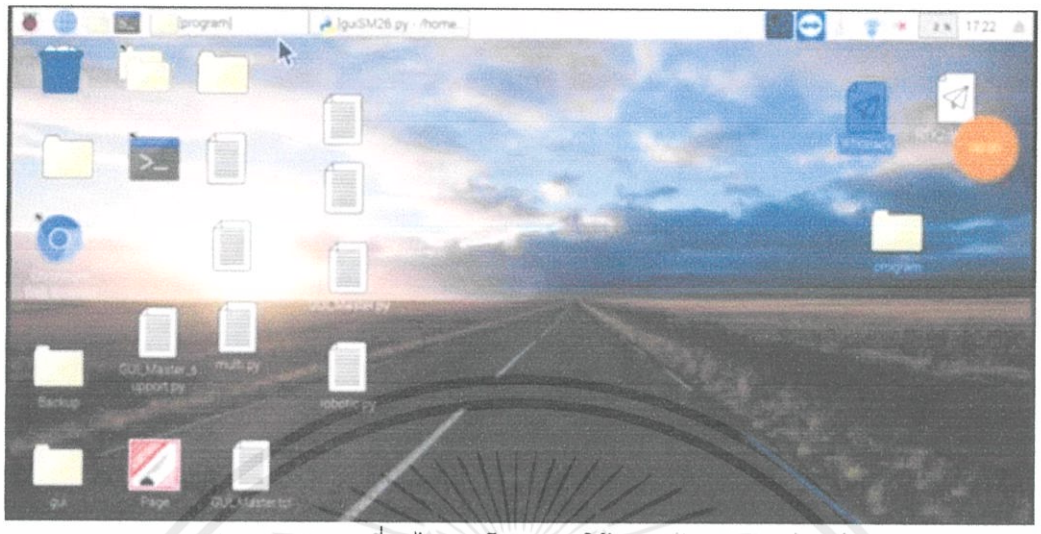
รูปที่ ๓ การจัดวางสายอากาศ และสถานีจลน์บนรถเกษตรกรรม

๒. ต่อสาย MICRO-USB และสาย HDMI ของจอแสดงผลเข้ากับพอร์ต USB และพอร์ต HDMI จากกล่องในรูปที่ ๒ แล้วทำการติดตั้งจอแสดงผลไว้บริเวณพวงมาลัยดังรูปที่ ๔

รูปที่ ๔ การติดตั้งจอแสเงผลบนรถเกษตรกรรม

๓. เปิดใช้งานสถานีจลน์โดยกดปุ่มเปิด-ปิดสวิตช์การทำงาน

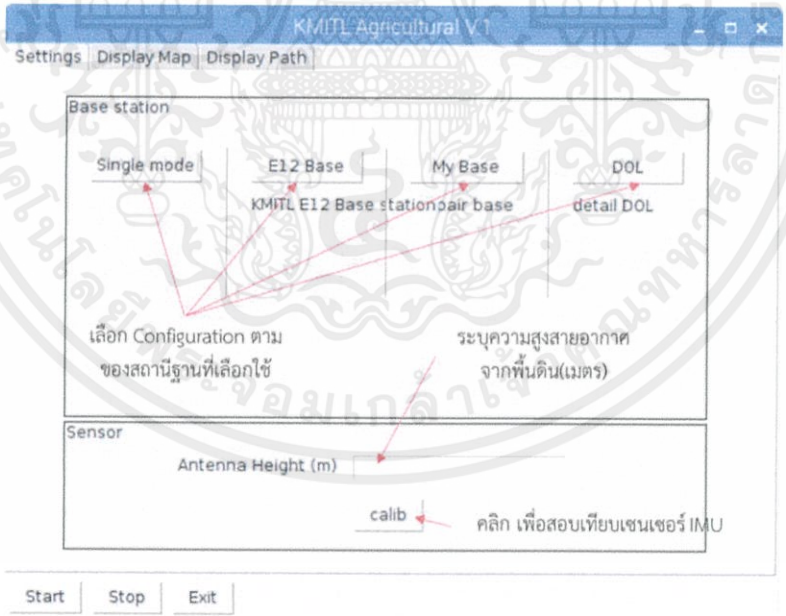
๔. หลังจาก RASPBERRY PI พร้อมใช้งานแล้ว จะปรากฏหน้าจอระบบปฏิบัติการณ์ขึ้นมา (เรียกว่า DESKTOP) จะพบโฟลเดอร์โปรแกรมใช้งานหลัก คือ PROGRAM ดังรูปที่ ๕



รูป ภาคผนวก ฉ ที่ 5 ไอคอนโปรแกรมใช้งานหลักของสถานีจลน์

๕. ดับเบิ้ลคลิกที่โปรแกรม RTKTRACK (ไอคอนที่ติดสัญลักษณ์สีฟ้าบนหน้าต่างการใช้งานหลัก) ตามรูปที่ ๕ แล้วจะพบหน้าต่างแสดงผล ตามรูปที่ ๖

๖. หลังจากเปิดโปรแกรมแล้ว จะพบกับหน้า “SETTINGS” ดังรูปที่ ๖

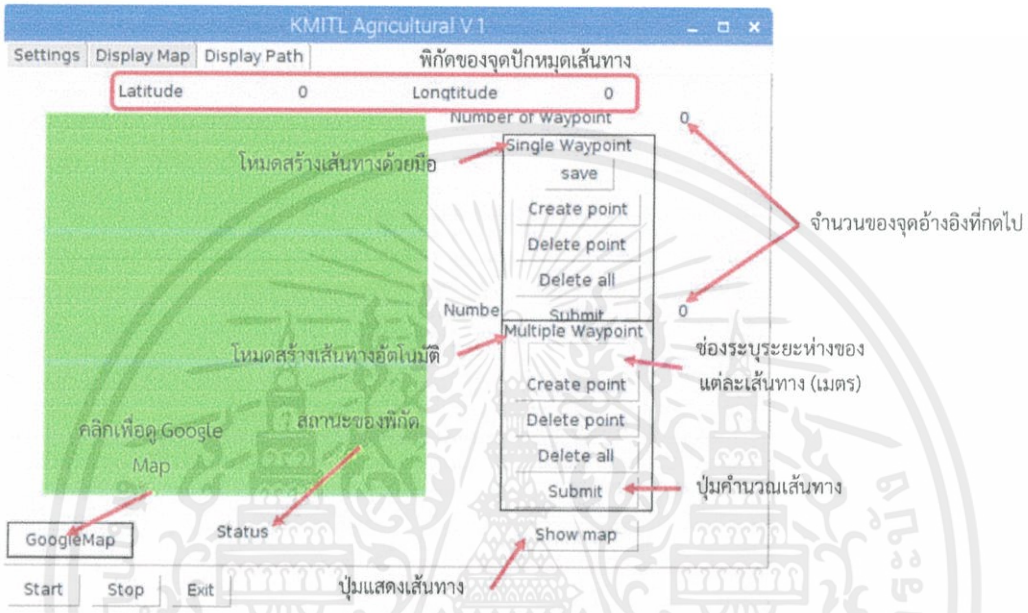


รูปที่ ๖ แถบ “SETTINGS”

จากรูปที่ ๖ จะประกอบด้วย ๒ ส่วนที่สำคัญ คือ ส่วนของการเลือกชนิดของสถานีฐานที่ต้องการเชื่อมต่อ (ในกรอบ BASE STATION) เพื่อใช้ระบุตำแหน่งร่วมกับเทคนิค RTK โดยการคลิกที่ปุ่ม SINGLE MODE จะเป็นการระบุตำแหน่งโดยไม่รับค่าแก้จากสถานีฐานใดๆ ซึ่งจะใช้ในกรณีที่ต้องการทราบตำแหน่งสถานีจรคร่าวๆ ส่วนการคลิกปุ่ม E12 BASE จะเป็นการรับค่าแก้จากสถานีฐานชนิดสองความถี่ที่ทำการติดตั้งเอง หรือของกรมผังเมือง ต่อมา หากคลิกที่ปุ่ม MY BASE จะเป็นการรับค่าแก้จากสถานีฐานชนิดหนึ่งความถี่ที่มาพร้อมกับชุดอุปกรณ์ฯ และปุ่มสุดท้ายจะใช้สำหรับการรับค่าแก้จากสถานีฐานของกรมที่ดิน การจะเลือกใช้สถานีฐานแต่ละประเภท จะต้องพิจารณาระยะเส้นหลักระหว่างสถานีฐานกับสถานีจลน์ ความเสถียรของสัญญาณอินเทอร์เน็ต และสภาพแวดล้อมรอบๆ สถานีจลน์อีกด้วย เพื่อให้การระบุตำแหน่งมีความต้อเนื่อง ละมีความแม่นยำสูง แล้วตั้งค่าในกรอบ SENSOR โดยการกรอกความสูงของสายอากาศจากพื้นดินในหน่วยเมตรลงในช่อง “ANTENNA HEIGHT (M)” เพื่อใช้ในการป้อนกลับความเอียง หากต้องการสอบเทียบเซนเซอร์ IMU สามารถคลิกที่ปุ่ม “CALIB” แล้วหมุนตัวเซนเซอร์นาน ๓๐ วินาที เพื่อวัดค่าสอบเทียบ จากนี้ไป จะเป็นการสร้างเส้นทางจราจรบนพื้นที่เพาะปลูก ดังหัวข้อที่ต่อไป

การสร้างเส้นทางเดินรถ

หลังจากเลือกสถานีฐานแล้ว ให้คลิกที่แถบ “Display Path” จะได้ปรากฏหน้าจอดังรูปที่ ๗ ให้คลิกที่ “Start” เพื่อเริ่มการระบุตำแหน่งโดยใช้เทคนิค RTK สำหรับสร้างจุดปักหมุดเส้นทางเดินรถ

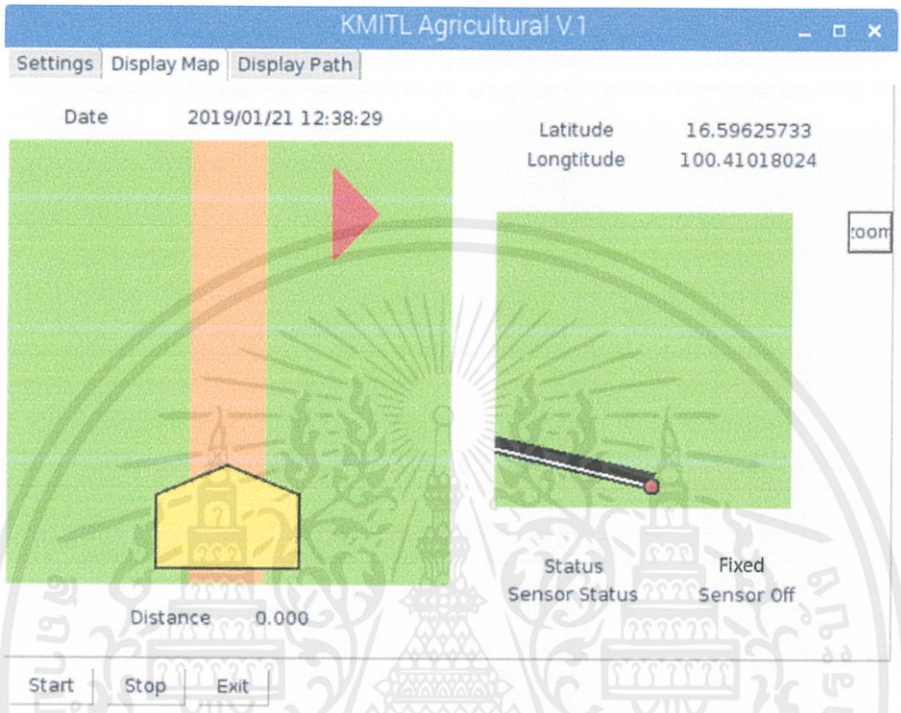


รูปที่ ๗ แถบ “Display Path”

จากรูปที่ ๗ จะบ่งบอกรายละเอียดของการใช้งานแถบนี้ ให้ผู้ขับขี่ คอยสังเกต Status เมื่อพบว่าพิกัดมีสถานะเป็น Fixed แล้ว จึงจะเริ่มปักหมุดเส้นทาง โดยโหมดการปักหมุดเส้นทางด้วยมือ ผู้ขับขี่จะต้องขับรถไปยังจุดที่จะปักหมุดเส้นทาง แล้วคลิกที่ “Create point” ในกรอบ “Single Waypoint” ทีละจุด หากมีการเก็บพิกัดผิดและต้องการเก็บใหม่ ให้คลิกที่ปุ่ม “Delete point” เพื่อลบจุดปักหมุดล่าสุด หรือคลิกที่ “Delete all” เพื่อลบทุกจุดแล้วเริ่มเก็บใหม่ทั้งหมด หลังจากเก็บจุดปักหมุดครบทั้งเส้นทางแล้ว ให้คลิกที่ “Submit” เพื่อคำนวณเส้นทาง ในกรณีที่เลือกใช้โหมดการสร้างเส้นทางแบบอัตโนมัติ ให้ผู้ขับขี่ ขับรถไปที่บริเวณขอบของพื้นที่เพาะปลูก แล้วทำการเก็บจุดปักหมุดรอบพื้นที่เป็นจำนวน ๔ จุด (จุดทั้ง ๔ มุมของพื้นที่เพาะปลูก) โดยคลิกที่ “Create point” ในกรอบ “Multi Waypoint” แล้วกรอกระยะห่างระหว่างเส้นทางในหน่วยเมตรลงไปในช่องว่าง คลิกที่ “Submit” จากนั้นคลิกที่ “Show map” เพื่อแสดงเส้นทางบนจอ หากผู้ขับขี่ต้องการดูแผนที่บน Google Maps สามารถคลิกที่ปุ่ม “GoogleMap” เพื่อแสดงแผนที่ดังกล่าวได้ หลังจากได้เส้นทาง ให้ผู้ขับขี่ขับรถไปยังจุดแรกของเส้นทาง เพิ่มเริ่มการนำทางรถ

การนำทางรถไถ หรือรถปลูก

ให้ผู้ขับขี่คลิกที่แถบ “DISPLAY MAP” จะได้ผลลัพธ์ดังรูปที่ ๘



รูปที่ ๘ แถบ “DISPLAY MAP”

จากรูปที่ ๘ เมื่อใช้งานชุดอุปกรณ์แล้ว ทำการสังเกตที่หน้าจอแสดงผลด้านซ้าย จะมีรูปห้าเหลี่ยม ซึ่งแสดงถึงตัวรถ และเส้นทางเดินรถจะถูกแทนด้วยแถบสีส้ม สำหรับการแจ้งเตือนระยะห่างและทิศทางที่รถเบี้ยวออกไปจากเส้นทางการจราจร ผู้ขับขี่สามารถสังเกตได้จาก “DISTANCE” ซึ่งจะบอกถึงระยะห่างจากตัวรถถึงเส้นทาจราจรในหน่วยเมตร และบอกทิศทางในการปรับตำแหน่งด้วยลูกศรสีแดง นอกจากนี้ จะบอกเวลาที่ทำการทดสอบไว้หลังคำว่า “DATE” เพื่อให้ผู้ขับสามารถนับเวลาที่ใช้ในการทำงานเพื่อการคำนวณค่าแรงภายหลังได้ ในส่วนของหน้าจอด้านขวา ยังมีการแสดงพิกัดของรถในรูปแบบ ๒ มิติ เพื่อให้ผู้ขับขี่ทราบว่าตนอยู่บริเวณใดของพื้นที่ทดสอบ โดยจะแทนพิกัดของรถด้วยจุดกลมสีแดง แทนเส้นทางจราจรด้วยเส้นสีดำ และจะใช้เส้นสีขาวแสดงแทนเส้นการจราจรปัจจุบันที่รถวิ่งอยู่ นอกจากนี้ จะแสดงตำแหน่งของรถในรูปแบบพิกัดภูมิศาสตร์ด้วยค่าละติจูดกับลองจิจูด ซึ่งอยู่หลังคำว่า “LATITUDE” กับ “LONGITUDE” ตามลำดับ หากผู้ขับขี่ต้องการขยายมุมมอง สามารถกดที่ปุ่ม “ZOOM” เพื่อปรับขนาดมุมมองได้ตามต้องการ ในส่วนของคำว่า “STATUS” จะบ่งบอกถึงสถานะคุณภาพพิกัดในขณะนั้น โดยพิกัดที่มีความแม่นยำสูง จะต้องมีสถานะ

เป็น FIXED เท่านั้น สุดท้าย ตรงคำว่า “SENSOR STATUS” จะบ่งบอกถึงความพร้อมในการใช้งานเซนเซอร์ IMU หากเซนเซอร์ฯ มีความพร้อมในการใช้งาน จะแสดงคำว่า “ON” ถ้าไม่พร้อมใช้งาน จะแสดงคำว่า “OFF” แทน ซึ่งทางผู้วิจัยจะใช้ในการป้อนกลับความเอียงของรถ และช่วยนำทางรถขณะที่ไม่สามารถรับสัญญาณจีเอ็นเอสเอสได้